Implementación del Análisis Estadístico en Ciencias Biomédicas mediante un Asistente Interactivo Multiagente basado en Modelos de Lenguaje Grande

Henry Joseth González Torres

Grupo de Investigación Innovación, Desarrollo, Biotecnología en Salud y Medio Ambiente

Universidad Simón Bolívar

Universidad del Valle

Posgrado en Ciencias Biomédicas / Escuela de Ciencias Básicas / Facultad de Salud

Universidad del Valle Santiago de Cali, Colombia

2025

Implementación del Análisis Estadístico en Ciencias Biomédicas mediante un Asistente Interactivo Multiagente basado en Modelos de Lenguaje Grande

Henry Joseth Gonzalez Torres

Tesis presentada como requisito parcial para optar al título de:

Doctor en Ciencias Biomédicas

Director:

Dr. Juvenal Yosa Reyes

Codirector:

Dr. Julio Cesar Montoya Villegas

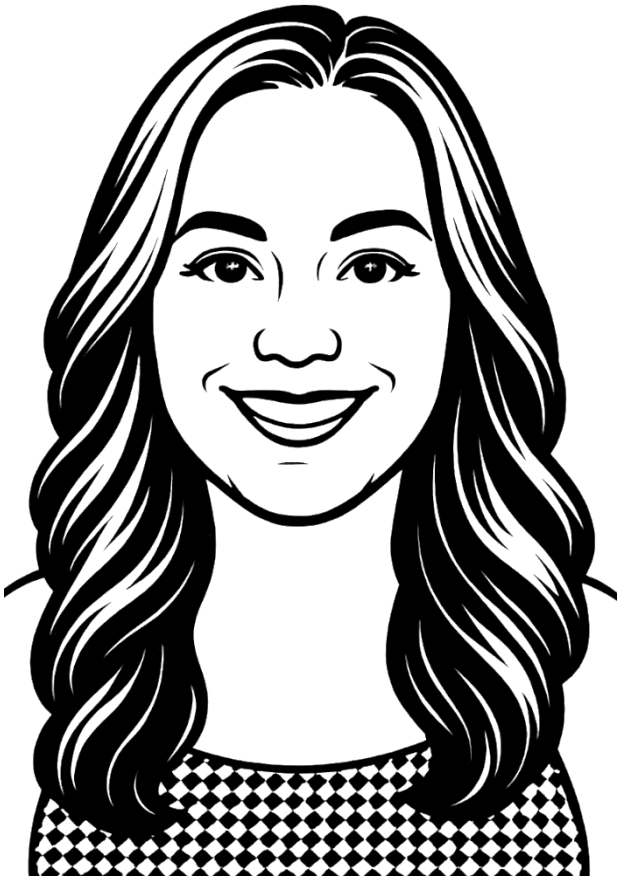
Universidad del Valle

Posgrado en Ciencias Biomédicas / Escuela de Ciencias Básicas / Facultad de Salud

Universidad del Valle Santiago de Cali, Colombia

2025

Dedicatoria



In memoriam a Adry Agamez Diaz

Oct.18.1989 – Ago.19.202

Agradecimientos

En primer lugar, agradezco a **Jehová Dios**, quien todo lo permite. Su presencia fue mi sostén en los momentos más oscuros, dándome la fortaleza espiritual y emocional necesaria para no rendirme cuando todo parecía perdido.

A mis padres, **Custodio González Ortega** y **Elieth Torres Sierra**, por su amor incondicional, fe incuestionable y apoyo constante. Nunca dejaron de creer en mí, aun cuando yo mismo dudaba, y fueron mi ancla en este largo camino. «*Un aplauso pa' mami y papi, porque en verdad rompieron*».

Al **Dr. Juvenal Yosa Reyes** y al **Dr. Julio César Montoya**, quienes no solo creyeron en esta idea desde su origen, sino que fueron piezas fundamentales para que este proyecto pudiera comenzar y, sobre todo, llegar a término. Sin su acompañamiento académico, técnico y humano, este trabajo jamás habría sido posible.

A la **Dra. Liliana Salazar Monsalve**, Directora del Postgrado en Ciencias Biomédicas de la Universidad del Valle, por brindarme la oportunidad de continuar este camino académico. Su decisión hizo posible retomar un proceso que, en algún momento, parecía definitivamente cerrado.

A **Silvia Aldana Pérez**, quien apareció en el momento más oscuro de mi vida. Cuando todo mi plan de vida se había derrumbado, no permitiste que me perdiera en la oscuridad. Me tomaste de la mano las veces que fue necesario, guiándome hacia la luz para poder llegar hasta aquí. Gracias por ser refugio, impulso y esperanza.

A mis hijos, **Elieth Yoveira González De Armas** y **Henry Joseth González Agamez** por ser bastión, motivación y propósito. Ustedes son la razón más profunda detrás de cada paso dado, de cada noche en vela y de cada decisión de seguir adelante.

Asimismo, agradezco a los **34 participantes** que participaron voluntariamente en la prueba de usabilidad, por brindar sus impresiones y enriquecer este trabajo con sus perspectivas desde la experiencia investigativa.

Finalmente, reconozco el respaldo institucional de la **Universidad Simón Bolívar – Sede Barranquilla, CO**, por facilitar los tiempos, medios académicos, tecnológicos y logísticos necesarios para la ejecución de este proyecto de investigación, especialmente a **Narledis Nuñez Bravo**, **Clara Inés Soto Aroca** y **Julieth Marcela Hollmann Mendoza** a quién tengo el honor de decirles amigas.

Resumen

Implementación del Análisis Estadístico en Ciencias Biomédicas mediante un Asistente Interactivo Multiagente basado en Modelos de Lenguaje Grande

El presente estudio tuvo como objetivo evaluar la implementación de un sistema multiagente interactivo basado en modelos de lenguaje grande (LLMs) para automatizar el análisis estadístico en investigaciones biomédicas, garantizando precisión, reproducibilidad y aplicabilidad práctica. Se desarrolló un asistente compuesto por siete agentes especializados —planificación, análisis, investigación contextual, generación, verificación, depuración y documentación— que colaboran de forma secuencial e iterativa con supervisión humana (Human-in-the-Loop). El diseño fue experimental in silico, utilizando datos reales y simulaciones en entorno controlado.

La validación funcional se llevó a cabo mediante una escala heurística de 15 ítems aplicada por expertos sobre 30 artículos científicos, evaluando precisión técnica, claridad del código, replicabilidad y documentación. Se obtuvo una alta concordancia interevaluador (AC1 = 0.831 ± 0.075) y una puntuación promedio de 4.38 ± 0.93. Destacaron la claridad del código (4.88 ± 0.33) y la documentación generada (4.88 ± 0.33). Sin embargo, se observaron limitaciones en la identificación automática de variables (1.85 ± 0.66) y la replicación exacta de resultados (3.33 ± 0.69). En paralelo, se evaluó la aplicabilidad práctica mediante la escala System Usability Scale (SUS) aplicada a 34 usuarios, complementada con análisis semántico-emocional de sus respuestas. El puntaje mediano de usabilidad fue de 78.8 (IQR: 72.5–87.5), con predominio de emociones positivas como confianza, anticipación y satisfacción.

Se concluye que el sistema multiagente mostró un desempeño funcional robusto y una buena aceptación por parte de los usuarios, posicionándose como una herramienta prometedora para fortalecer la reproducibilidad y eficiencia en el análisis estadístico aplicado a la ciencia biomédica.

Palabras clave: Modelos de lenguaje grande, agentes inteligentes, análisis estadístico automatizado, reproducibilidad científica, investigación biomédica, validación funcional, usabilidad.

Abstract

Implementation of an Interactive Multi-Agent Assistant Based on Large Language Models for Statistical Analysis in Biomedical Research

This study aimed to evaluate the implementation of an interactive multi-agent system based on Large Language Models (LLMs) to automate statistical analysis in biomedical research, ensuring precision, reproducibility, and practical applicability. The assistant was composed of seven specialized agents—planning, analysis, contextual research, generation, verification, debugging, and documentation—that collaborate sequentially and iteratively under a Human-in-the-Loop approach. The design was experimental in silico, using real biomedical data within a controlled simulation environment.

Functional validation was conducted using a 15-item ad hoc heuristic scale, applied by expert evaluators to outputs generated from 30 scientific articles. The evaluation focused on technical accuracy, code clarity, reproducibility, and quality of documentation. Results showed high inter-rater agreement (AC1 = 0.831 ± 0.075) and an overall functional score of 4.38 ± 0.93. Highest ratings were observed in code clarity (4.88 ± 0.33) and documentation quality (4.88 ± 0.33), while lower performance was found in automatic variable identification (1.85 ± 0.66) and full result replication (3.33 ± 0.69). Additionally, practical applicability was assessed using the System Usability Scale (SUS) completed by 34 users, alongside semantic-emotional analysis of their written feedback. The median usability score was 78.8 (IQR: 72.5–87.5), with a predominance of positive emotions such as trust, anticipation, and satisfaction.

The system demonstrated robust functional performance and favorable user reception, positioning it as a promising tool to enhance reproducibility and efficiency in statistical analysis for biomedical research.

Keywords: Large language models, intelligent agents, automated statistical analysis, scientific reproducibility, biomedical research, functional validation, usability.

Contenido

Pág.

Resumen VI

Abstract VII

Lista de figuras 11

Lista de tablas 13

Lista de cuadros 14

Lista de Ecuaciones 15

Introducción 17

Capítulo 1: Planteamiento de la Investigación 20
1.1 Planteamiento del Problema 20
1.2 Justificación de la Investigación 22
1.3 Objetivos de la Investigación 23
1.3.1 Objetivo General 23
1.3.2 Objetivos Específicos 24

Capítulo 2: Marco Conceptual 25
2.1 Aprendizaje Clásico (Classical Learning) 25
2.1.1 Aprendizaje No Supervisado (Unsupervised Learning) 25
2.1.1.1 Clustering (Agrupamiento) 26
2.1.1.2 Búsqueda de Patrones (Pattern Search) 27
2.1.1.3 Reducción de Dimensión (Dimension Reduction - Generalización) 29
2.1.2 Aprendizaje Supervisado 30
2.1.2.1 Clasificación (Classification) 30
2.1.2.2 Regresión (Regression) 31
2.2 Métodos de Ensamble (Ensemble Methods) 32
2.2.1 Bagging (Bootstrap Aggregating) 33
2.2.2 Boosting (Impulso) 33
2.2.3 Stacking (Apilamiento) 34
2.3 Redes Neuronales y Aprendizaje Profundo (Neural Nets and Deep Learning). 36
2.3.1 Perceptrones (MLP - Perceptrón Multicapa) 36
2.3.2 Autoencoders (Codificadores Automáticos) 37

2.3.2.1 Seq2Seq (Secuencia a Secuencia)....39
2.3.2.2 Redes Generativas Antagónicas (GAN - Generative Adversarial Networks)....39
2.3.2.3 Redes Neuronales Recurrentes (RNN - Recurrent Neural Networks)....41
2.3.2.4 LSM (Liquid State Machines)....43
2.3.2.5 LSTM (Long Short-Term Memory - Memoria a Largo Corto Plazo)....44
2.3.2.6 GRU (Gated Recurrent Unit - Unidad Recurrente Gated)....45
2.3.2.7 Redes Neuronales Convolucionales (CNN - Convolutional Neural Networks)....45
2.3.2.8 DCNN (Deep Convolutional Neural Networks - Red Neuronal Convolucional Profunda)....46
2.4 Aprendizaje por Refuerzo (Reinforcement Learning)....47
2.4.1 Algoritmos Genéticos....48
2.4.2 Q-Learning....49
2.4.2.1 Deep Q-Network (DQN)....50
2.4.2.2 SARSA (State-Action-Reward-State-Action)....51
2.4.2.3 A3C (Asynchronous Advantage Actor-Critic)....52
2.5 Evolución de los Modelos de Lenguaje Grande (LLM)....53
2.5.1 Transformers: La Arquitectura Base de los Modelos de Lenguaje Grande....54
2.5.2 LLMs y Agentes Inteligentes....56
2.5.2.1 Características y Funcionamiento Fundamental de un Agente....58
2.5.2.2 Clasificación de los Agentes Inteligentes....58
2.5.2.3 Interacción entre los Modelos de Lenguaje Grande (LLMs) y los Agentes Inteligentes....59
2.5.3 RAG y Agentes Inteligentes Basados en LLMs: Sinergia para el Análisis Automatizado....60
2.5.4 Aplicaciones de los LLMs y Agentes en la Investigación Biomédica....62
2.6 Estado del Arte o Antecedentes....64

Capítulo 3: Diseño Metodológico....67
3.1 Tipo de Estudio....67
3.2 Procedimiento....67
3.2.1 Desarrollo de Agentes Especializados....67
3.2.2 Integración del Modelo de Lenguaje Generativo....69
3.2.3 Métodos de Evaluación....71
3.2.4 Consideraciones Éticas....76

Capítulo 4: Resultados....77
4.1 Desarrollo del Asistente Interactivo Multiagente para Análisis Estadístico Automatizado....77
4.1.1 Arquitectura General del Sistema Multi-Agente....77
4.1.2 Componentes Clave y Flujo Algorítmico....78
4.1.2.1 Agente Planificador (Planner Agent)....79
4.1.2.2 Agente Analizador (Analyzer Agent)....83
4.1.2.3 Agente Investigador (Researcher Agent)....85
4.1.2.4 Agente Codificador (Coder Agent)....88
4.1.2.5 Agente Verificador (Verifier Agent)....91
4.1.2.6 Agente Depurador (Debugger Agent)....93
4.1.2.7 Agente Documentador (Documenter Agent)....95

4.1.2.8 Gestión del Conocimiento y Bases de Datos Vectoriales 98
4.1.2.9 El Rol Crítico del HITL 101
4.2 Evaluación heurística de cumplimiento funcional Asistente Multiagente IA 102
4.3 Evaluación de la Aplicabilidad y Usabilidad del Asistente en Contextos Reales de Investigación Biomédica 107
4.3.1 Evaluación Cuantitativa - System Usability Scale (SUS) 111
4.3.2 Análisis Semántico-Emocional de la Percepción del Usuario 113
4.3.2.1 Análisis Semántico-Emocional de la Percepción del Usuario – Eficacia 113
4.3.2.2 Análisis Semántico-Emocional de la Percepción del Usuario – Eficiencia 114
4.3.2.3 Análisis Semántico-Emocional de la Percepción del Usuario – Satisfacción 115
4.3.2.4 Valoración Global de Asistente Multiagente 116
4.3.3 Evaluación en la Contribución a la Co-creación de Conocimiento Científico .. 117

Capítulo 5: Discusiones 127

Capítulo 6: Conclusiones y recomendaciones 135
6.1 Conclusiones 135
6.2 Recomendaciones 137

Referencias Bibliográficas 139

A. Anexo: Instrumento de Evaluación Heurística–Funcional 153

B. Anexo: Evaluación de Usabilidad del Agente de Análisis Estadístico con LLMs 155

C. Anexo: Continuación del Cuadro 13.– Representación esquemática del plan de investigación por el Agente Investigador 159

D. Anexo: Continuación del Cuadro 14.– Inicialización automatizada del entorno computacional por el Agente Codificador 161

E. Anexo: Continuación del Cuadro 16.– Documentación estructural inicial del script: objetivos, funciones y estructura funcional 165

F. Anexo: Resultados gráficos y tabulares generados por el Asistente Multiagente IA en R 169

Lista de figuras

Pág.
Figura 1.– Agrupamiento no supervisado con datos sintomáticos de ejemplo. 26
Figura 2– Transformación de datos masivos en decisiones mediante análisis estructurado. 28
Figura 3.– Comparación entre PCA y LDA en la reducción de dimensión: varianza vs. separación de clases. 29
Figura 4.– Representación del proceso de clasificación automática. 30
Figura 5.– Regresiones en aprendizaje automático 32
Figura 6.– Esquema del método Bagging en aprendizaje automático. 33
Figura 7.– Representación del enfoque Boosting. 34
Figura 8.– Proceso de Stacking en aprendizaje de ensamble. 35
Figura 9.– Representación esquemática de un percentron 37
Figura 10.– Arquitectura de un autoencoder. 38
Figura 11.– Aprendizaje secuencia a secuencia con un codificador RNN y un decodificador RNN 39
Figura 12.– Arquitectura de una Red Generativa Antagónica (GAN). 40
Figura 13.– Arquitectura de una red neuronal recurrentes (RNN). 42
Figura 14.– Representación de una Liquid State Machine (LSM). 43
Figura 15.– Arquitectura interna de una celda LSTM. 44
Figura 16.– Arquitectura interna de una unidad GRU (Gated Recurrent Unit). 45
Figura 17.– Arquitectura de una Red Neuronal convolucional (CNN) 46
Figura 18.– Flujo de procesamiento en una Red Neuronal Convolucional Profunda (DCNN). 47
Figura 19.– Representación de un algoritmo genético. 48
Figura 20.– Ciclo de actualización del algoritmo Q-Learning. 50
Figura 21.– Esquema de Deep Q-Learning (DQN). 51
Figura 22.– Ciclo de interacción agente-entorno y Cadena de decisiones en aprendizaje por refuerzo – SARSA 52
Figura 23.– Esquema del modelo Actor-Critic en aprendizaje por refuerzo. 53
Figura 24.– Arquitectura del modelo Transformer. 55
Figura 25.– Estructura funcional de un agente inteligente basado en LLM. 57
Figura 26.– Arquitectura del enfoque RAG (Retrieval-Augmented Generation) con modelo de lenguaje grande (LLM) 61

Figura 27.– Coeficiente de concordancia AC1 de Gwet por criterio evaluado en la escala heurística 104
Figura 28 .– Desempeño promedio por criterio en la evaluación heurística funcional del Asistente Multiagente IA 105
Figura 29.- Red semántico-emocional de la dimensión «Eficacia». 114
Figura 30.- Red semántico-emocional de la dimensión «Eficiencia» 115
Figura 31.- Red semántico-emocional de la dimensión “Satisfacción”. 116
Figura 32.- Red semántico-emocional Global. 117

Lista de tablas

Pág.

Tabla 1.– Evaluación heurística funcional del Asistente Multiagente IA basada en la concordancia entre evaluadores y desempeño 103
Tabla 2.– Perfil de los participantes según nivel de formación académico..... 108
Tabla 3.– Nivel de competencia en software estadístico (SPSS, STATA, R y Python) por formación académica..... 109
Tabla 4.– Frecuencia de uso de análisis estadístico y experiencia previa con herramientas de inteligencia artificial según nivel académico..... 110
Tabla 5.– Resultados de la evaluación a través del System Usability Scale (SUS) 112

Lista de cuadros

Pág.
Cuadro 1.- Punto de entrada del algoritmo principal..... 79
Cuadro 2.- Algoritmo del Agente Planificador: Análisis y Generación de Subtareas..... 82
Cuadro 3.- Algoritmo del Agente Analizador: Evaluación de Necesidad de Conocimiento Externo 85
Cuadro 4.- Algoritmo del Agente Investigador: Síntesis de Información Contextual 88
Cuadro 5.- Algoritmo del Agente Codificador: Generación de Código y Dependencias... 91
Cuadro 6.- Algoritmo del Agente Depurador: Diagnóstico y Corrección de Código 93
Cuadro 7.- Algoritmo del Agente Verificador: Generación de Pruebas y Retroalimentación 95
Cuadro 8.- Algoritmo del Agente Documentador: Generación de Documentación Técnica y Funcional 98
Cuadro 9.- Algoritmo de Q&A: Pregunta sobre documentación/código 98
Cuadro 10.- Instrucción semántica inicial para la para la co-creación con el Asistente Interactivo Multiagente LLM de flujo de trabajo analítico en R 118
Cuadro 11.– Descomposición funcional de tareas por el Agente Planificador-Analizador 119
Cuadro 12.– Heurística inferencial del Agente Planificador: estructura de datos, transformaciones y flujos de análisis..... 120
Cuadro 13.– Representación esquemática del plan de investigación por el Agente Investigador 121
Cuadro 14.– Inicialización automatizada del entorno computacional por el Agente Codificador 122
Cuadro 15.– Evaluación de la calidad metodológica del script por el Agente Documentador 123
Cuadro 16.– Documentación estructural inicial del script: objetivos, funciones y estructura funcional 124

Lista de Ecuaciones

Pág.
Ecuación 1.– ecuación de actualización de Bellman 49
Ecuación 2.– ecuación de actualización de SARSA 52
Ecuación 3.– Formula de la distancia euclidiana entre dos vectores 86
Ecuación 4.– Similitud del coseno entre dos vectores 97
Ecuación 5.– Fórmula general de actualización iterativa en aprendizaje automático 97

Introducción

La creciente complejidad de los análisis estadísticos en investigación biomédica, sumada a la necesidad de garantizar reproducibilidad, precisión y trazabilidad metodológica, ha planteado nuevos desafíos para investigadores y equipos clínicos que no siempre cuentan con formación avanzada en estadística o programación [1]. En este contexto, los avances en inteligencia artificial, particularmente los modelos de lenguaje grande (Large Language Models, LLMs), han abierto un abanico de oportunidades para automatizar tareas técnicas complejas, incluyendo el análisis estadístico, sin renunciar a la supervisión humana ni a la rigurosidad científica.

Los LLMs, como GPT-4, han demostrado capacidad para interpretar, sintetizar y generar contenido técnico con coherencia semántica y precisión estructural. Sin embargo, su aplicación directa a flujos analíticos complejos como los que se desarrollan en contextos biomédicos exige arquitecturas más sofisticadas que superen la generación textual autónoma y monolítica. En ese sentido, el presente estudio propone la construcción e implementación de un asistente multiagente interactivo, compuesto por una serie de agentes especializados en tareas específicas del análisis estadístico, que colaboran bajo un esquema coordinado y supervisado por el usuario (Human-in-the-Loop). Este modelo plantea una solución estructurada y trazable para automatizar con control experto la planificación, ejecución, verificación y documentación de análisis estadísticos en el ámbito biomédico.

Diversos estudios han abordado la integración de LLMs en ciencia de datos, destacando su potencial para generar código funcional, interpretar lenguaje natural, y asistir en tareas de preprocesamiento, modelado y visualización. Sin embargo, la mayoría de estas aplicaciones son unipersonales, orientadas a tareas aisladas y carecen de mecanismos de validación integrada, memoria de largo plazo o interacción modular entre procesos. Autores

como Gao et al. (2024) [2] y Chiarello et al. (2024)[3] han señalado que el uso efectivo de LLMs en entornos científicos requiere no solo mejoras algorítmicas, sino también estructuras de control y supervisión que garanticen calidad, coherencia y responsabilidad técnica.

En la literatura biomédica, el problema de la baja reproducibilidad y la dificultad para replicar resultados estadísticos ha sido ampliamente documentado. Iniciativas como el movimiento Open Science y el uso de reportes reproducibles en R o Python buscan contrarrestar estas debilidades, pero requieren competencias que no todos los equipos clínicos o investigadores poseen. Por tanto, el desarrollo de asistentes inteligentes que automaticen el análisis con criterios de rigor metodológico representa un paso innovador en la democratización de la analítica en salud.

Este estudio se propuso evaluar la implementación de un asistente interactivo multiagente basado en LLMs, diseñado para automatizar integralmente el análisis estadístico en investigaciones biomédicas, manteniendo altos estándares de precisión, replicabilidad y utilidad documental. La motivación central es responder a la necesidad de herramientas que reduzcan la carga técnica, mejoren la calidad del análisis y fomenten la participación activa de investigadores no especialistas en estadística o programación.

El enfoque metodológico fue de tipo experimental *in silico*, con orientación aplicada y carácter cuantitativo. Se desarrolló un prototipo funcional compuesto por siete agentes (planificador, analista, investigador, generador de código, verificador, depurador y documentador), cada uno implementado mediante una instancia de LLM configurada con system prompts especializados. El sistema fue sometido a un proceso iterativo de validación funcional utilizando 30 artículos científicos como fuente de solicitudes analíticas reales.

La validación se realizó a través de una escala heurística funcional de 15 ítems aplicada por expertos, complementada con pruebas de usabilidad y análisis semántico-emocional de las percepciones de 34 usuarios durante sesiones de interacción con el asistente. Se emplearon métodos cuantitativos para la estimación de concordancia (coeficiente AC1), análisis descriptivos e inferenciales y procesamiento textual con lexicones en español.

El presente estudio no busca evaluar exhaustivamente la totalidad de capacidades de los LLMs, sino validar el funcionamiento integrado de un sistema multiagente orientado al análisis estadístico aplicado. Su alcance se limita al dominio de la investigación biomédica, en tareas relacionadas con la planificación, ejecución y documentación de modelos analíticos clásicos (regresión, ANOVA, modelos de supervivencia, entre otros). No se evaluó la interacción con datos clínicos en tiempo real, ni su integración con sistemas hospitalarios o plataformas de inteligencia empresarial.

Entre las principales limitaciones se encuentra la dependencia del desempeño del modelo base (GPT-4), la variabilidad contextual de los prompts, y el hecho de que la validación se realizó en entorno simulado. Sin embargo, estos factores fueron mitigados mediante la intervención humana supervisora, ciclos iterativos de prueba y control estructurado de los flujos entre agentes.

La propuesta de un asistente multiagente basado en LLMs representa una innovación metodológica y tecnológica en el campo de la bioestadística aplicada. Al combinar automatización con supervisión humana, se ofrece una solución de alto valor para centros de investigación, instituciones clínicas y grupos académicos que requieren análisis estadístico riguroso sin contar con personal especializado en programación.

Este enfoque contribuye a reducir las barreras técnicas, incrementar la eficiencia en la producción de resultados y mejorar la trazabilidad del razonamiento analítico, fortaleciendo los principios de ciencia abierta y reproducible. Además, sienta las bases para futuras aplicaciones en contextos clínicos reales, y para el desarrollo de sistemas híbridos que integren procesamiento de lenguaje natural, razonamiento estadístico y aprendizaje adaptativo.

Capítulo 1: Planteamiento de la Investigación

1.1 Planteamiento del Problema

La investigación biomédica ha enfrentado al desafío crítico de garantizar la reproducibilidad y la transparencia en el análisis de datos, pilares fundamentales para el avance del conocimiento científico [4]. A pesar de los avances en herramientas estadísticas y computacionales, la complejidad inherente de los análisis y la dependencia de habilidades avanzadas en programación y estadística limitan el acceso de investigadores a métodos de análisis robustos y replicables [5]. Esto ha resultado en una brecha significativa entre la producción de datos experimentales y su correcta interpretación [6,7], lo que compromete la reproducibilidad de los resultados y, en consecuencia, la confianza en las conclusiones científicas [8,9].

Es impactante encontrar que el 49% de las investigaciones publicadas en Ciencias Biomédicas no pueden ser replicadas directamente, y que 75% de estos nunca pudieron ser reproducidos, incluso con el apoyo de los investigadores o correcciones en el método descrito [10]. Asunto que no es de menor importancia, dado por el impacto que tiene en la inyección de recursos al sistema de ciencia y tecnología, el cual es menor año tras año, en algunos países como Colombia y Chile para el 2023 disminuyó del 15% y para el 2024 fue de alrededor de un 34% [11,12], y que cuya reducción presupuestaría esta fundada en que muchos de los recursos destinados tanto a ciencias básicas como “aplicadas”, tienen una baja reproducibilidad [13,14] lo cual impacta directamente en la credibilidad asociada a la ciencia per se [15,16].

Una solución a esta problemática es la integración hombre-maquina, a través de la utilización de machine learning (ML), lo que no es solo una opción viable, sino, sensata ya que permite incluso crear modelos futuros, a partir de información recopilada y utilizarlos en la consolidación de la información para tomar decisiones a partir del procesamiento de dicha

información [17–19]. Actualmente, hay diferentes tipos de herramientas de análisis estadístico y de generación de texto a partir del procesamiento de información, que ayuda a los investigadores a analizar situaciones y proponer soluciones o ideas innovadoras [20–22]

La disponibilidad de Modelos de Lenguaje Grande (LLMs) ha abierto nuevas posibilidades en el ámbito del análisis automatizado de datos, al permitir el procesamiento de lenguaje natural y la generación de código de manera eficiente y precisa [4]. Estos modelos, entrenados con vastas cantidades de información, tienen el potencial de democratizar el acceso a herramientas analíticas avanzadas al permitir que investigadores, incluso con conocimientos limitados en programación, realicen análisis estadísticos complejos de manera autónoma [23,24]. Sin embargo, su implementación en el contexto de la investigación biomédica plantea interrogantes críticas sobre su precisión, reproducibilidad y aplicabilidad práctica en escenarios reales [25].

Para poder implementar más eficientemente la IA en investigación, es importante desarrollar agentes que no solo sean capaces de realizar análisis precisos, sino también de garantizar la replicabilidad de los resultados en diversos contextos [26–28]. Evaluando su funcionalidad en escenarios prácticos, incluyendo datos generados directamente en laboratorios biomédicos, para asegurar su utilidad y accesibilidad como herramientas en el entorno académico y científico.

El desarrollo y la validación de estos agentes representan una oportunidad para transformar la forma en que se realizan y comunican los análisis estadísticos en el ámbito biomédico, promoviendo la reproducibilidad científica y mejorando la eficiencia en la interpretación de datos experimentales [9,29]. Sin embargo, para que estos agentes sean aceptados y ampliamente utilizados, es necesario evaluar si son capaces de replicar con precisión los resultados obtenidos con métodos tradicionales, así mismo en su efectividad en escenarios prácticos donde los datos provienen de entornos experimentales complejos [30,31].

El presente estudio busca abordar desarrollar, evaluar y validar la implementación de agentes LLMs para la automatización del análisis estadístico en investigaciones biomédicas, se planteó como pregunta de investigación ¿Puede un Asistente Interactivo

Multiagente basado en LLMs integrarse de forma efectiva y válida en flujos de análisis estadístico biomédico, aportando a la generación de conocimiento científico bajo criterios de reproducibilidad, precisión y aplicabilidad en entornos reales de investigación?

1.2 Justificación de la Investigación

En la actualidad hay un consenso claro que la IA y sus campos, entre ellos y los LLMs son herramientas potentes que potencian el trabajo científico, especialmente en tareas repetitivas, análisis de datos y generación de hipótesis; Sin embargo, la creatividad, la ética y la toma de decisiones críticas siguen siendo dominios exclusivamente humanos y es poco probable que sean reemplazado por algún sistema artificial [32,33].

La problemática de la reproducibilidad de los ensayos clínicos es real y una manera de garantizar la reproducibilidad es objetivizar el diseño de los experimentos, ya que muchas veces el diseño de los mismo se ve afectado por el nivel de experticia del investigador [34,35], fuera de los problemas propios de una investigación, tales como limitaciones en el presupuesto o consecución de la muestra, eventos que se deben tener en cuenta cuando se ajusta un experimento a condiciones reales. Este tipo de estudios como el aquí planteado brinda herramientas para facilitar al investigador el diseño final de su experimento [36,37].

Por otra parte, el desconocimiento de técnicas estadísticas para el ajuste, normalización o imputación de datos hace que no se les saque el máximo provecho a los datos obtenidos, muchas veces quedándose en el análisis principal, sin profundizar en el mismo, ya que se requieren conocimientos especializados que se obtienen a través del tiempo [38]; siendo este punto uno de los aportes más significativos de esta propuesta ya que pone a disposición un pool de agentes, que asisten al investigador en el manejo de información, dándole la oportunidad de aprovechar mejor los datos obtenidos en el laboratorio o en la revisión de información de los pacientes [39].

En este sentido, el uso de los LLMs ha sido fundamental en la producción de nuevo conocimiento, ejemplo de ello es PubMedGTP [40] que es capaz de hacer búsqueda de artículos y hacer resúmenes precisos sobre un tema en particular; o BioBERT [41] que

permite hacer minería de texto conectándose a PubMed; ese es el verdadero impacto de los LLMs en las Ciencias Biomédicas , reducen el tiempo para analizar y procesar grandes volúmenes de información; no limitándose a texto, sino que pueden analizar grandes conjuntos de datos estadísticos, resumiendo patrones y generando informes interpretables [42] o escribiendo códigos en R, Python o SAS para realizar análisis estadísticos complejos [43], estas razones hacen necesario seguir desarrollando este tipo de agentes pero con un enfoque más al análisis de datos y que entren a ayudar a los investigadores en su producción de nuevo conocimiento.

La implementación de agentes basados en LLMs representa un paso significativo hacia la democratización del conocimiento y la optimización del análisis estadístico en ciencias biomédicas. Estas herramientas no solo promueven la reproducibilidad científica al estandarizar procesos complejos, sino que también potencian la capacidad de los investigadores para explorar y aprovechar al máximo los datos generados, superando barreras de experticia y recursos limitados [44]. Al integrar la inteligencia artificial en la investigación biomédica, se abren nuevas oportunidades para mejorar la calidad, eficiencia y alcance de los estudios, fortaleciendo la producción de conocimiento científico con un enfoque ético y colaborativo que complemente la creatividad y el juicio humano, razón que al final persigue esta investigación.

1.3 Objetivos de la Investigación

1.3.1 Objetivo General

Evaluar la implementación de un Asistente Interactivo Multiagente basado en Modelos de Lenguaje Grande (LLMs) en el desarrollo de análisis estadísticos en investigaciones biomédicas, bajo un enfoque Human-in-the-Loop (HITL), con el fin de valorar su capacidad para aportar conocimiento replicable, preciso y aplicable en contextos reales del ámbito biomédico.

1.3.2 Objetivos Específicos

- Desarrollar un conjunto de agentes especializados basados en modelos de lenguaje (LLMs) que operen de forma colaborativa como un asistente interactivo para apoyar el análisis estadístico estructurado en investigaciones biomédicas.

- Evaluar la funcionalidad técnica y metodológica del asistente multiagente en términos de precisión, reproducibilidad y validez analítica, a partir de su aplicación en escenarios de investigación con datos reales.

- Validar la aplicabilidad práctica del sistema desde la perspectiva de su utilidad científica, eficacia operativa y experiencia de usuario, considerando su potencial para contribuir de manera significativa a la producción estructurada de conocimiento biomédico.

Capítulo 2: Marco Conceptual

El aprendizaje automático (Machine Learning, ML) es un campo de la inteligencia artificial que permite a las computadoras aprender patrones a partir de datos sin necesidad de una programación explícita. Su objetivo es mejorar el rendimiento de un sistema en una tarea específica mediante la experiencia adquirida a través de datos. Se clasifica en diferentes enfoques, cada uno con metodologías y aplicaciones particulares [45,46].

2.1 Aprendizaje Clásico (Classical Learning)

El aprendizaje clásico es el conjunto de métodos tradicionales de ML, basado en el análisis estadístico y la extracción de patrones en conjuntos de datos estructurados. Se divide en aprendizaje supervisado y aprendizaje no supervisado, dependiendo de si los datos de entrenamiento incluyen etiquetas o no [45,46].

2.1.1 Aprendizaje No Supervisado (Unsupervised Learning)

El aprendizaje no supervisado es un enfoque en el que los modelos identifican patrones en los datos sin tener etiquetas predefinidas. Su propósito es encontrar estructuras ocultas en los datos y organizarlos en grupos o categorías. Es especialmente útil cuando no se dispone de información previa sobre los datos, y su aplicación se encuentra en la segmentación de clientes, reducción de dimensiones y detección de anomalías [47,48].

2.1.1.1 Clustering (Agrupamiento)

El clustering es una técnica fundamental en el aprendizaje no supervisado que agrupa objetos similares en conjuntos homogéneos. Su objetivo es maximizar la similitud dentro de un mismo grupo y minimizar la similitud entre diferentes grupos. Se usa en áreas como la segmentación de mercado, la detección de comunidades en redes sociales y la clasificación de imágenes médicas. Existen diferentes tipos de algoritmos para agrupación clúster, los más comunes son: DBSCAN, Mean-Shift, K-Means, Agrupamiento jerárquicoy Fuzzy C-Means [49]

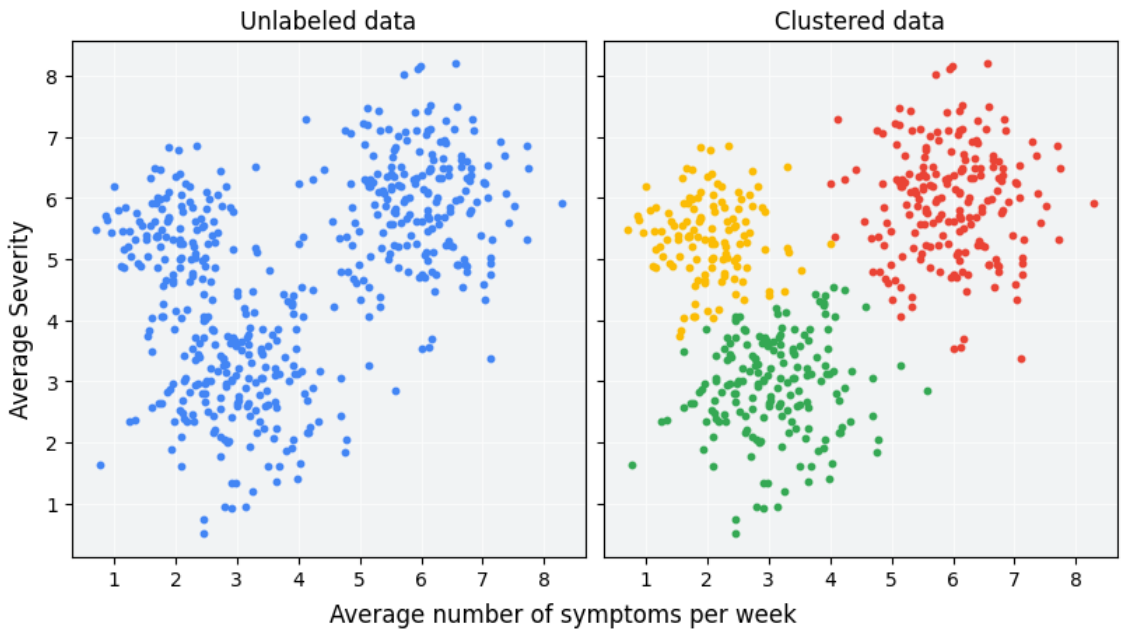


Figura 1.– Agrupamiento no supervisado con datos sintomáticos de ejemplo.

Ejemplo de análisis no supervisado mediante clustering: el panel izquierdo muestra datos sin etiquetar, mientras que el derecho presenta agrupamientos detectados automáticamente según la severidad y frecuencia promedio de síntomas, revelando estructuras latentes en los datos. Tomado de: https://developers.google.com/machine-learning/clustering/overview?hl=es-419

El algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise) es un método de agrupamiento basado en la densidad que identifica regiones densas de puntos de datos y las separa de áreas dispersas o con ruido. Se utiliza ampliamente en la detección de anomalías y agrupamiento de datos geoespaciales. A diferencia de K-Means, no requiere definir un número de grupos previamente, lo que lo hace útil en aplicaciones donde la estructura de los datos no es clara [50,51].

El algoritmo Mean-Shift busca densidades de puntos en un espacio de características y agrupa los datos en torno a estas áreas densas. Se basa en la idea de mover iterativamente los puntos de datos hacia la región de mayor densidad hasta que converjan en un centro común. Es muy usado en segmentación de imágenes y análisis de movimiento en visión computacional [50,52].

El algoritmo K-Means es uno de los más utilizados en clustering. Funciona dividiendo los datos en k grupos, donde k es un número predefinido. Los datos se agrupan en función de la distancia a los centros de cada clúster, que se actualizan iterativamente hasta lograr una convergencia. Se usa en segmentación de clientes y compresión de datos [50,53,54].

El Agrupamiento jerárquico es una técnica que construye una estructura en forma de árbol para organizar los datos en grupos de manera sucesiva. Se parte de cada punto como un clúster individual y, en cada iteración, los clústeres más cercanos se combinan hasta formar un solo grupo. Es útil cuando no se conoce el número óptimo de clústeres en los datos [50].

El algoritmo Fuzzy C-Means es una variante del clustering tradicional donde un punto de datos puede pertenecer a múltiples clústeres con diferentes grados de pertenencia. Se usa en sistemas difusos y aplicaciones de clasificación en imágenes médicas y procesamiento de señales [50,55].

2.1.1.2 Búsqueda de Patrones (Pattern Search)

La búsqueda de patrones es una técnica utilizada para encontrar asociaciones y relaciones en grandes volúmenes de datos. Su aplicación principal es en la minería de datos, detección de fraudes y análisis de comportamiento del consumidor [56,57].

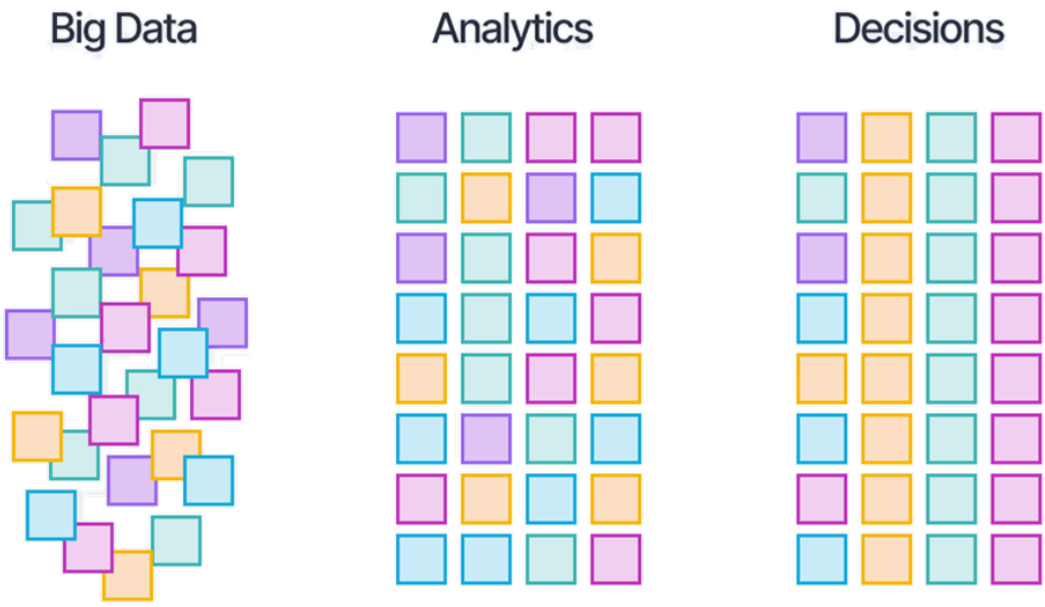


Figura 2– Transformación de datos masivos en decisiones mediante análisis estructurado.

Los datos desordenados provenientes de fuentes masivas (Big Data) se organizan mediante procesos analíticos (Analytics) hasta convertirse en información estructurada que facilita la toma de decisiones (Decisions). Tomado de: https://www.v7labs.com/blog/pattern-recognition-guide

El algoritmo Euclat es un método eficiente para la búsqueda de patrones frecuentes en grandes bases de datos. Se basa en la intersección de conjuntos de elementos para encontrar combinaciones repetitivas en los datos. El algoritmo Apriori es uno de los más utilizados en minería de datos para descubrir reglas de asociación entre diferentes elementos en bases de datos transaccionales. Se usa comúnmente en análisis de cestas de compra para determinar qué productos suelen comprarse juntos. El algoritmo FP-Growth es una versión optimizada de Apriori que utiliza una estructura de árbol de patrones frecuentes para acelerar el proceso de descubrimiento de asociaciones en grandes conjuntos de datos [56,57].

2.1.1.3 Reducción de Dimensión (Dimension Reduction - Generalización)

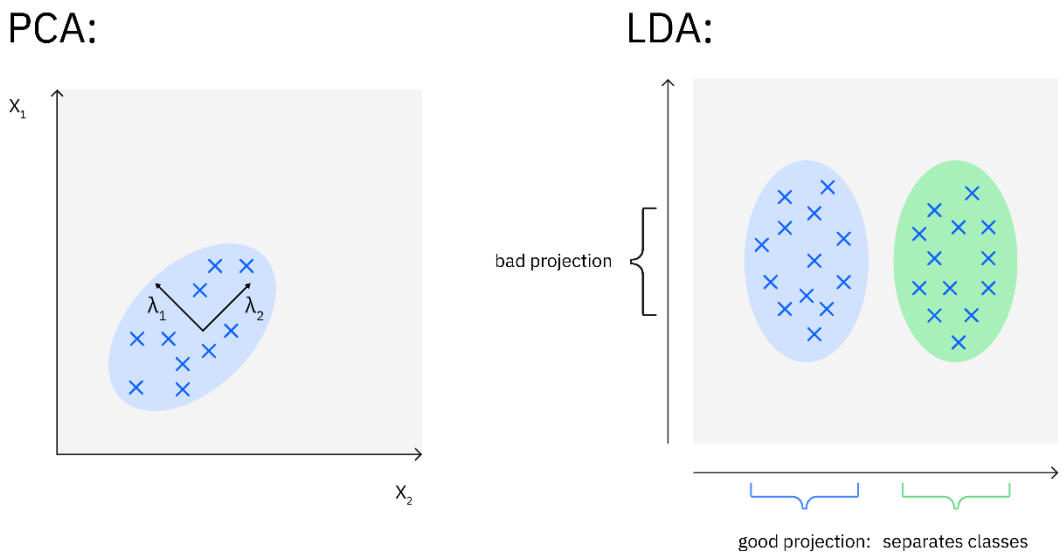


Figura 3.– Comparación entre PCA y LDA en la reducción de dimensión: varianza vs. separación de clases.

Representación teórica de PCA y LDA: PCA proyecta los datos maximizando la varianza total sin considerar etiquetas de clase, mientras que LDA busca una proyección óptima que maximice la separabilidad entre clases. Tomado de: https://www.ibm.com/es-es/think/topics/dimensionality-reduction

Los métodos de reducción de dimensión permiten simplificar conjuntos de datos al eliminar variables redundantes, mejorando la eficiencia y la capacidad de generalización de los modelos. Las más utilizadas son: t-SNE, PCA, LSA, SVD y LDA [57].

El t-SNE (t-Distributed Stochastic Neighbor Embedding) es una técnica de reducción de dimensión utilizada para visualizar datos de alta dimensión en un espacio de dos o tres dimensiones. Es particularmente útil en el análisis de datos genéticos y la exploración de estructuras de datos [58]. El PCA (Principal Component Analysis) es un método estadístico que transforma un conjunto de variables en un conjunto de componentes principales ortogonales. Se usa ampliamente en el análisis de imágenes, compresión de datos y eliminación de ruido [59]. El LSA (Latent Semantic Analysis) se emplea en el procesamiento de lenguaje natural para encontrar relaciones ocultas entre términos en grandes volúmenes de texto, se utiliza en motores de búsqueda y análisis de documentos [60]. El SVD (Singular

Value Decomposition) es una técnica algebraica para descomponer matrices y extraer características principales. Se usa en la compresión de imágenes y en la reducción de ruido en señales. El LDA (Linear Discriminant Analysis) es una técnica supervisada utilizada para la reducción de dimensiones en problemas de clasificación, optimizando la separación de clases en el espacio de características [57].

2.1.2 Aprendizaje Supervisado

El aprendizaje supervisado es un enfoque donde el modelo aprende a partir de datos etiquetados, es decir, cada entrada de datos tiene una salida conocida. El objetivo es que el modelo generalice y pueda hacer predicciones sobre nuevos datos. Se divide en dos categorías principales, dependiendo de si la salida es discreta o continua: clasificación y regresión [45,61,62].

2.1.2.1 Clasificación (Classification)

La clasificación es un tipo de problema supervisado donde el objetivo es asignar una etiqueta o categoría a una entrada de datos. Se usa en reconocimiento de imágenes, detección de spam y diagnóstico médico [61,63].

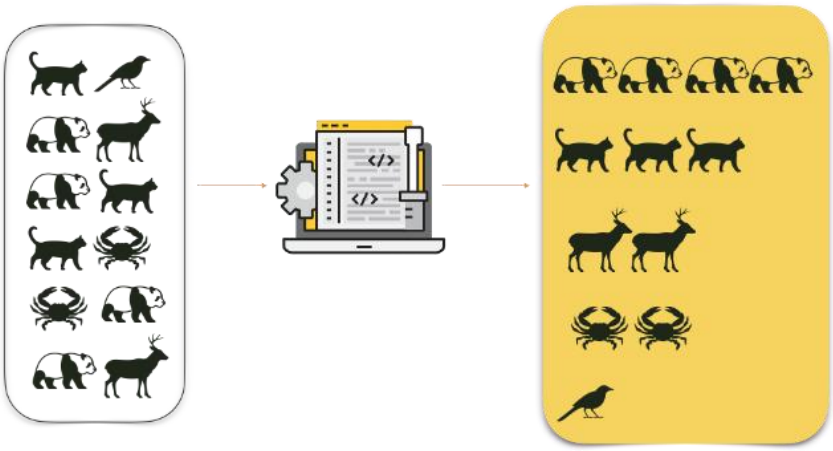


Figura 4.– Representación del proceso de clasificación automática

Un algoritmo de clasificación procesa datos no etiquetados (a la izquierda) y los organiza en grupos homogéneos (a la derecha), asignando automáticamente una categoría a cada elemento según sus similitudes. Tomado de: https://aprendeia.com/2018/03/22/clasificacion-de-machine-learning/

El algoritmo Naive Bayes se basa en el Teorema de Bayes, que calcula la probabilidad de que una instancia pertenezca a una clase dada la evidencia observada. Se asume que las características son independientes entre sí, lo cual no siempre es cierto, pero funciona bien en problemas como clasificación de texto y filtrado de spam debido a su eficiencia y velocidad. El K-Nearest Neighbors (K-NN) es un algoritmo basado en la proximidad, donde una nueva instancia se clasifica según la mayoría de sus K vecinos más cercanos en el espacio de características. Se usa en reconocimiento de patrones y sistemas de recomendación, aunque su rendimiento disminuye en conjuntos de datos grandes [61,64].

Las Support Vector Machines (SVM) buscan encontrar un hiperplano óptimo que separe las diferentes clases en el espacio de datos. Son eficaces en conjuntos de datos de alta dimensión y se aplican en biometría, clasificación de imágenes y análisis de sentimientos. Los árboles de decisión son estructuras jerárquicas que toman decisiones basadas en reglas de sí/no. Son fáciles de interpretar y se usan en diagnóstico médico, análisis de riesgo financiero y estrategias de marketing. Aunque su nombre sugiere regresión, este modelo se usa en clasificación binaria al predecir probabilidades entre dos clases. Se aplica en diagnóstico de enfermedades, predicción de abandono de clientes y modelado de riesgos [61].

2.1.2.2 Regresión (Regression)

La regresión es un tipo de aprendizaje supervisado donde la salida es un valor continuo en lugar de una categoría. Se utiliza en predicción de precios, estimación de ventas y análisis económico.

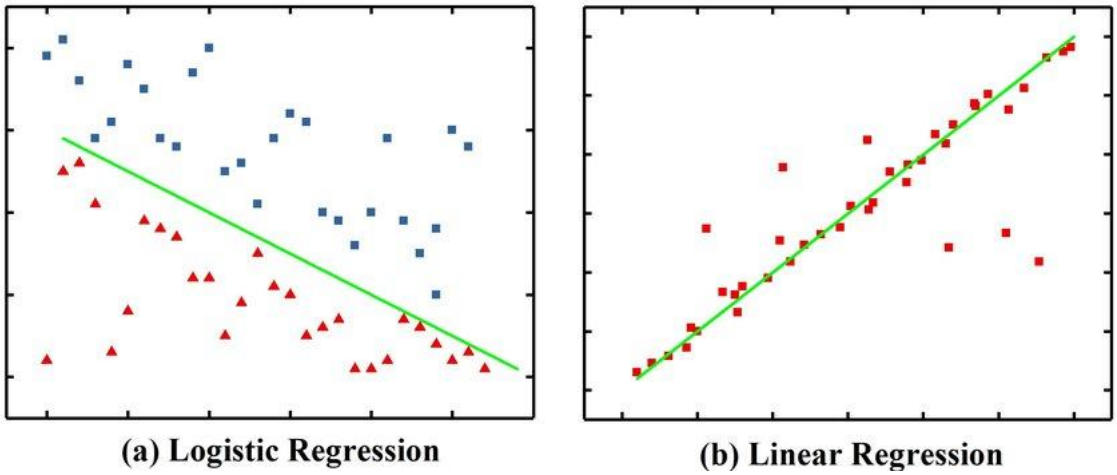


Figura 5.– Regresiones en aprendizaje automático

Comparación de dos enfoques del aprendizaje supervisado: (a) la regresión logística, utilizada para clasificación binaria, y (b) la regresión lineal, usada para predecir valores continuos. Tomado de: https://www.researchgate.net/figure/Logistic-regression-and-linear-regression_fig1_335786324

La regresión lineal modela la relación entre una variable dependiente y una o más variables independientes mediante una línea recta. Es útil en predicción de precios inmobiliarios, demanda de productos y análisis de tendencias. También se encuentra la regresión polinómica que es un caso especial de la regresión lineal al modelar relaciones no lineales, utilizando polinomios de mayor grado. Se usa en análisis de curvas de crecimiento y modelado de fenómenos físicos; así mismo esta la regresión Ridge/Lasso, en ambas se regulariza la regresión lineal, en ambos casos se penalizan los coeficientes para evitar el sobreajuste. Ridge utiliza una penalización L2, mientras que Lasso usa una penalización L1 para reducir características irrelevantes.

2.2 Métodos de Ensamble (Ensemble Methods)

Los métodos de ensamble que combinan múltiples modelos de aprendizaje para mejorar la precisión y reducir la varianza; mejorando la precisión, reducir el sobreajuste y aumentar la generalización en la predicción. En lugar de depender de un solo modelo, los métodos de ensamble agrupan diversos algoritmos para producir un resultado más robusto y confiable [45,62].

2.2.1 Bagging (Bootstrap Aggregating)

El bagging es un método que entrena múltiples modelos sobre subconjuntos de datos creados mediante muestreo con reemplazo (bootstrap). Luego, sus predicciones se combinan mediante votación (para clasificación) o promediado (para regresión). Ayuda a reducir la varianza y mejorar la estabilidad del modelo. El Random Forest es un método que construye múltiples árboles de decisión y promedia sus predicciones. Es robusto contra el sobreajuste y se usa en diagnóstico médico y clasificación de imágenes [61,62].

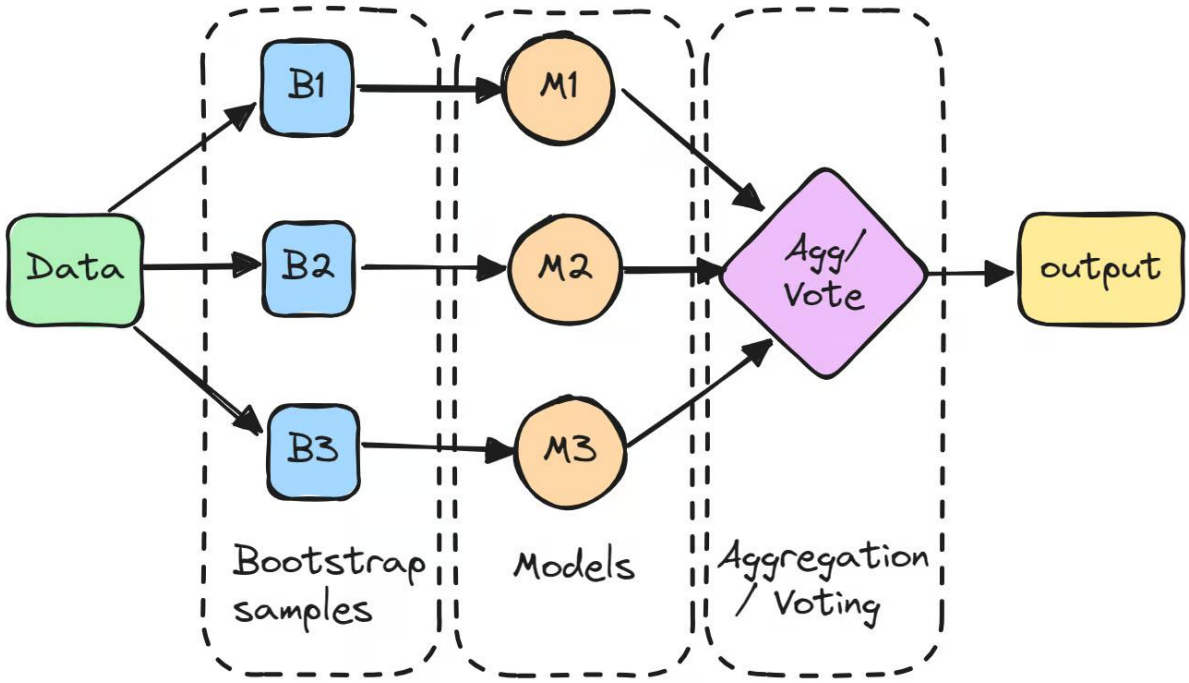


Figura 6.– Esquema del método Bagging en aprendizaje automático.

Método Bagging genera múltiples subconjuntos de datos (bootstrap), entrena modelos independientes sobre ellos y luego agrega sus predicciones mediante votación para obtener una salida final más robusta. Tomado de: https://www.datacamp.com/tutorial/what-bagging-in-machine-learning-a-guide-with-examples

2.2.2 Boosting (Impulso)

A diferencia del bagging, el boosting entrena modelos secuenciales, el boosting construye modelos secuencialmente, donde cada nuevo modelo corrige los errores cometidos por el anterior. A diferencia del bagging, que entrena modelos en paralelo, el boosting da más peso a las observaciones mal predichas en cada iteración [61,62].

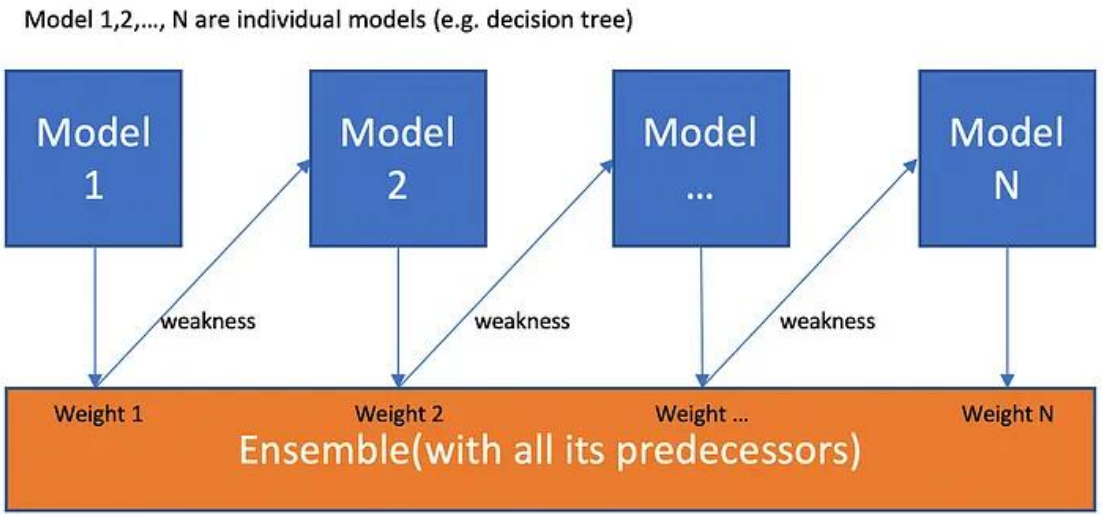


Figura 7.– Representación del enfoque Boosting.

El principio del Boosting, una técnica de aprendizaje de ensamble en la que múltiples modelos débiles (por ejemplo, árboles de decisión) se entrenan de manera secuencial. Cada nuevo modelo se enfoca en corregir los errores o "debilidades" del anterior, y todos los modelos se combinan mediante un sistema de ponderación para formar un modelo final más preciso y robusto. Tomado de: https://medium.com/@dishantkharkar9/about-boosting-and-gradient-boosting-algorithm-98dd4081ec18

El algoritmo XGBoost está altamente optimizado y su principal uso es la predicción de enfermedades y modelos de crédito. El algoritmo AdaBoost puede diferenciar las instancias mal clasificadas y las refuerza en la siguiente iteración; se usa en detección de objetos y clasificación de clientes. El CatBoost, optimiza la clasificación en conjuntos de datos con muchas variables categóricas, usado en finanzas y comercio electrónico. Y el LightGBM tiene un algoritmo rápido y eficiente que se usa en análisis de series temporales y modelado de riesgos financieros [61,62].

2.2.3 Stacking (Apilamiento)

El Stacking (Apilamiento) es una técnica avanzada de métodos de ensamble, utilizada para mejorar la precisión de los modelos en Machine Learning. A diferencia de Bagging y Boosting, que combinan modelos de la misma familia o de características similares, el Stacking permite combinar diferentes tipos de modelos, como árboles de decisión, redes

neuronales, regresión logística y máquinas de soporte vectorial, para construir un predictor más robusto [62,65].

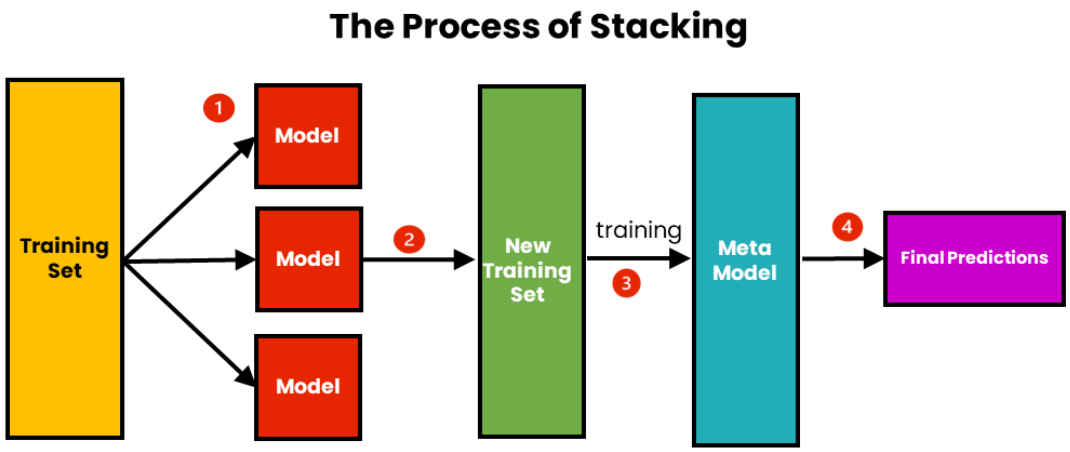


Figura 8.– Proceso de Stacking en aprendizaje de ensamble.

En el stacking múltiples modelos base se entrenan con el conjunto original y sus predicciones se combinan para formar un nuevo conjunto de entrenamiento, que alimenta a un modelo meta encargado de generar las predicciones finales. Tomado de: https://medium.com/@brijesh_soni/stacking-to-improve-model-performance-a-comprehensive-guide-on-ensemble-learning-in-python-9ed53c93ce28

Este algoritmo sigue una estructura de múltiples capas donde los modelos base generan predicciones y un modelo meta-aprende a combinarlas. Su arquitectura se compone de un entrenamiento de modelos base los cuales pueden incluir árboles de decisión, redes neuronales, SVM, regresiones lineales, entre otros; paso siguiente es la generación de predicciones por cada modelo base, la cual es independiente, sobre los datos de prueba; se crean subconjuntos de datos donde las características son las predicciones de los modelos base. Y por último se entrena el meta-modelo, al igual que una regresión logística, Random Forest o una red neuronal simple, toma como entrada las predicciones generadas por los modelos base y aprende cómo combinarlas de manera óptima. De este modo se busca minimizar los errores de los modelos base y generar una predicción final más precisa. Predicción Final [62,65,66].

2.3 Redes Neuronales y Aprendizaje Profundo (Neural Nets and Deep Learning)

El aprendizaje profundo (Deep Learning) es un subcampo del aprendizaje automático (Machine Learning) que utiliza redes neuronales artificiales para modelar y aprender patrones complejos en los datos. A diferencia de los modelos de aprendizaje automático tradicionales, las redes neuronales profundas pueden procesar grandes volúmenes de información y aprender representaciones jerárquicas a través de múltiples capas ocultas [45,61].

Estas técnicas han revolucionado aplicaciones en visión por computadora, procesamiento del lenguaje natural, reconocimiento de voz, generación de contenido y bioinformática. Los avances en hardware, como las unidades de procesamiento gráfico (GPU) y los TPU (Tensor Processing Units), han permitido entrenar modelos cada vez más grandes y complejos, como los Modelos de Lenguaje Grande (LLMs) [45,67].

Las principales arquitecturas dentro del aprendizaje profundo incluyen Perceptrones Multicapa (MLP), Autoencoders, Redes Generativas Antagónicas (GAN), Redes Neuronales Recurrentes (RNN) y Redes Neuronales Convolucionales (CNN) [45,61].

2.3.1 Perceptrones (MLP - Perceptrón Multicapa)

El Perceptrón Multicapa (MLP) es el modelo fundamental de las redes neuronales artificiales y representa una extensión del Perceptrón simple, incorporando múltiples capas de neuronas que permiten aprender patrones más complejos en los datos. Los MLP son considerados redes neuronales de alimentación hacia adelante (feedforward neural networks), ya que la información fluye en una sola dirección, desde la entrada hasta la salida, sin ciclos ni retroalimentación entre las neuronas. Son ampliamente utilizados en clasificación, regresión, predicción de series temporales y diagnóstico médico [68].

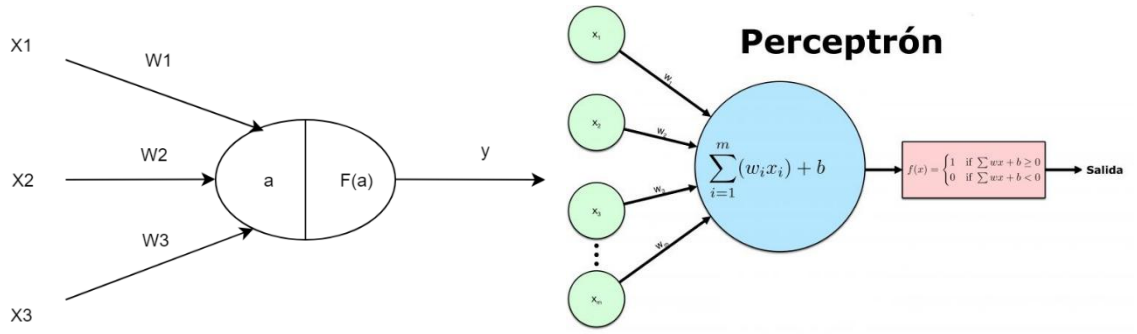


Figura 9.– Representación esquemática de un percentron

Un perceptrón, la unidad básica de una red neuronal: cada entrada x_i es ponderada por un peso w_i, se calcula una suma ponderada más un sesgo b, y se aplica una función de activación para generar una salida binaria y.Tomado de: https://blog.josemarianoalvarez.com/2018/06/10/el-perceptron-como-neurona-artificial/

La estructura de un MLP, se compone de tres tipos de capas principales: 1) Capa de Entrada, la cual recibe los datos de entrada en forma de vectores numéricos. Esta capa cada neurona representa una característica o variable del conjunto de datos. 2) Capas Ocultas, constituyen las entre la entrada y la salida, son las que aplican transformaciones matemáticas mediante funciones de activación para extraer patrones de los datos. Entre más capas ocultas tenga el modelo, más profundo será su aprendizaje (Deep Learning). Y 3) la capa de Salida, esta produce la predicción final basada en los cálculos de las capas ocultas, dependiente del problema así será su función de activación, i.e., en un problema de clasificación, esta capa usará una función de activación como Softmax o Sigmoide, pero, si el problema es de regresión, puede utilizar una función de activación lineal [68,69].

2.3.2 Autoencoders (Codificadores Automáticos)

Los Autoencoders son un tipo de red neuronal utilizada en aprendizaje no supervisado. Su objetivo principal es aprender representaciones compactas y significativas de los datos originales sin necesidad de etiquetas. Se emplean en tareas como reducción de dimensionalidad, detección de anomalías, generación de datos sintéticos y compresión de datos. Los autoencoders funcionan de manera similar a los modelos de compresión de datos, donde la red neuronal aprende a representar la información en una versión más compacta y luego intenta reconstruir los datos originales a partir de esa representación comprimida [70].

En general un Autoencoders tiene un funcionamiento general, basado en dos redes neuronales que trabajan en conjunto, un Codificador (Encoder), que reduce la dimensionalidad de los datos de entrada, extrae las características más relevantes de la información original y transforma los datos en un espacio latente de menor dimensión. Un espacio latente (Latent Space), que es la representación intercepta del conjunto de datos, básicamente es la información esencial del input en un formato más compacto y la segunda neurona que sería el Decodificador (Decoder), que intenta reconstruir los datos originales a partir del espacio latente y aprende a generar una versión lo más cercana posible al input original [70,71].

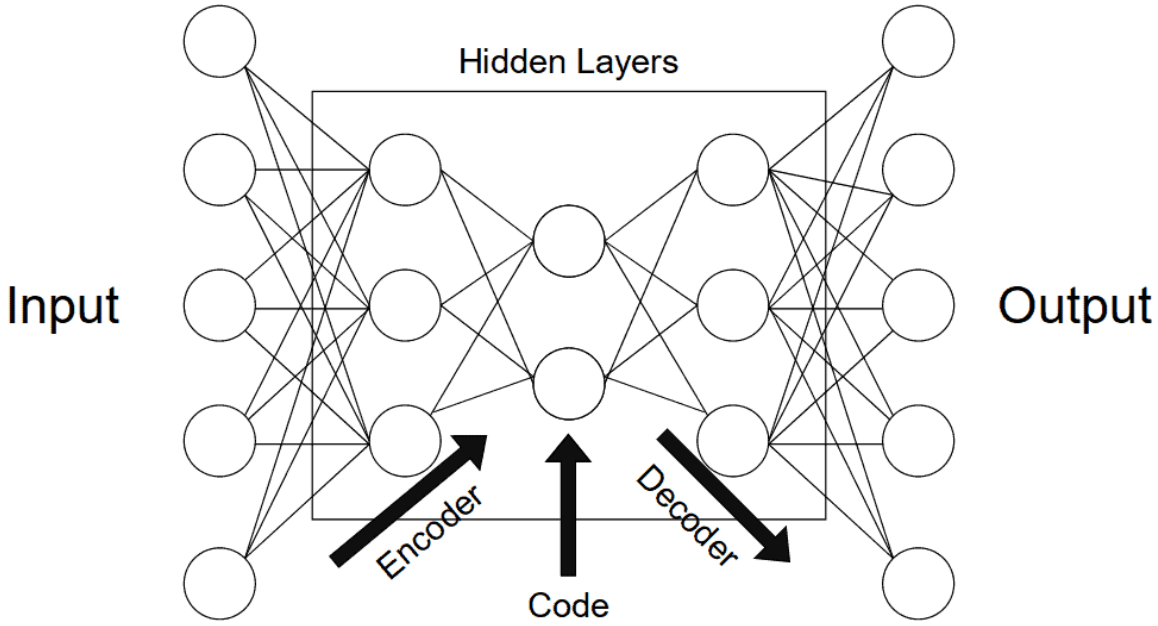


Figura 10.– Arquitectura de un autoencoder.

Autoencoder típico con capas de entrada, codificación (encoder), representación intermedia (code), decodificación (decoder) y salida. Su objetivo es aprender una representación comprimida de los datos para luego reconstruirlos, útil en reducción de dimensión, detección de anomalías o compresión.

Entrada → Codificador → Espacio Latente → Decodificador → Salida Reconstruida

Dado que son redes diseñadas para la reducción de dimensión y detección de anomalías, estos están diseñados para minimizar la diferencia entre la entrada y la salida reconstruida, utilizando una función de pérdida basada en la diferencia entre ambas [70,71].

2.3.2.1 Seq2Seq (Secuencia a Secuencia)

El modelo Sequence-to-Sequence (Seq2Seq) es una arquitectura basada en Autoencoders diseñada para procesamiento de secuencias, utilizada en traducción automática, generación de texto, chatbots y resúmenes de documentos. Estos tiene una estructura que consta de dos redes neuronales recurrentes (RNN), 1) el Codificador (Encoder), el cual toma la secuencia de entrada (por ejemplo, una oración en inglés) y la transforma en una representación latente compacta, suele utilizar una arquitectura como LSTM (Long Short-Term Memory) o GRU (Gated Recurrent Unit) para manejar dependencias a largo plazo. 2) Decodificador (Decoder), que usa la representación del codificador para generar la secuencia de salida (por ejemplo, la traducción al español); en el ejemplo, predice palabra por palabra basándose en los estados ocultos de la RNN. Y tenemos un mecanismo de Atención (Attention Mechanism), que permite al decodificador enfocarse en diferentes partes de la secuencia de entrada en cada paso. Este es muy útil en traducción automática y modelos de generación de texto avanzados [72].

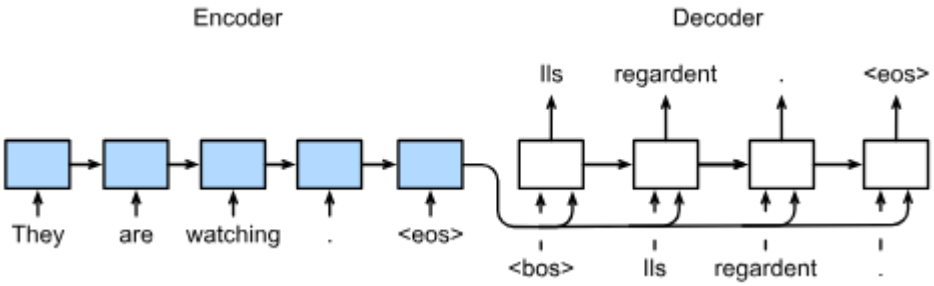


Figura 11.– Aprendizaje secuencia a secuencia con un codificador RNN y un decodificador RNN

Esquema de codificador-decodificador donde una secuencia de entrada es procesada por el encoder y transformada en un vector de contexto, que luego es utilizado por el decoder para generar la secuencia de salida, comúnmente aplicado en tareas de traducción automática o resumen de texto. Tomado de: https://d2l.ai/chapter_recurrent-modern/seq2seq.html

2.3.2.2 Redes Generativas Antagónicas (GAN - Generative Adversarial Networks)

Las Redes Generativas Antagónicas (GANs) representan un enfoque innovador dentro del aprendizaje profundo para la generación de datos sintéticos. Introducidas por Ian Goodfellow en 2014, las GANs han revolucionado múltiples áreas de la inteligencia artificial,

en especial en el ámbito de la visión computacional, síntesis de imágenes y generación de contenido multimedia [73].

El principio fundamental detrás de las GANs radica en la competencia entre dos redes neuronales profundas, conocidas como el generador y el discriminador, que interactúan en un marco de entrenamiento adversarial. Este modelo de competencia se asemeja a un juego de suma cero, donde el generador intenta crear datos sintéticos realistas, mientras que el discriminador busca diferenciar entre los datos generados y los datos reales del conjunto de entrenamiento [73].

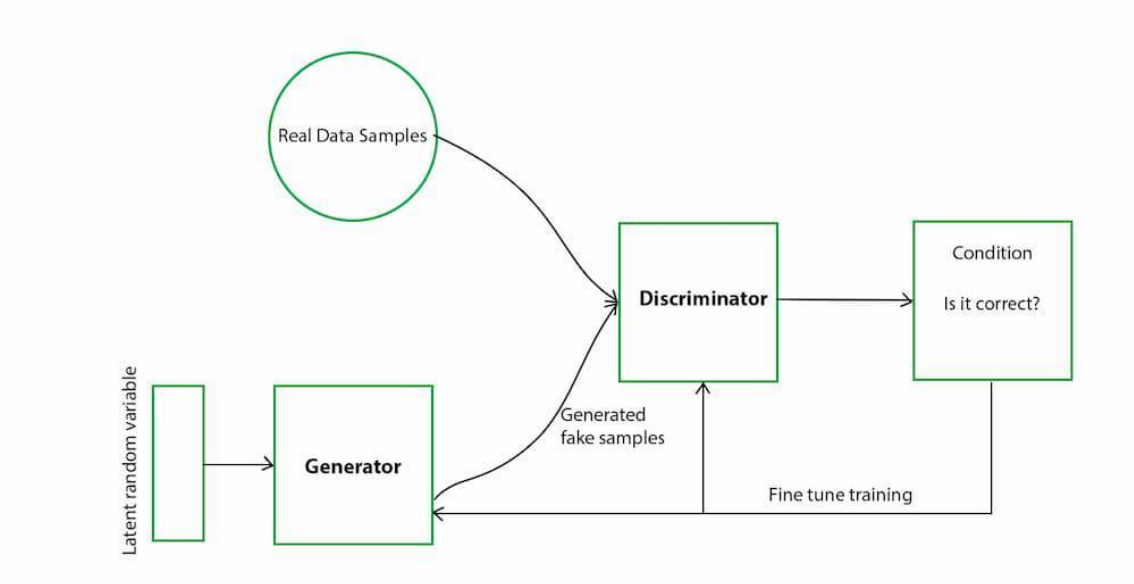


Figura 12.– Arquitectura de una Red Generativa Antagónica (GAN).

Representación del funcionamiento de una GAN, compuesta por un generador, que crea muestras sintéticas a partir de ruido, y un discriminador, que intenta distinguir entre datos reales y generados. Ambos modelos se entrenan en competencia, mejorando mutuamente hasta que el generador produce datos indistinguibles de los reales. Tomado de: https://www.geeksforgeeks.org/generative-adversarial-network-gan/

Durante el proceso de entrenamiento, el generador mejora continuamente su capacidad para engañar al discriminador, y este último, a su vez, mejora su precisión en la identificación de datos falsos. A medida que la GAN se entrena, el generador produce datos sintéticos cada vez más convincentes, hasta el punto en que el discriminador no puede distinguir entre los datos reales y los generados [73,74].

Las GANs han sido aplicadas en una amplia variedad de campos, desde la generación de imágenes realistas, como las caras sintéticas creadas por modelos como StyleGAN. La reconstrucción y mejora de imágenes, utilizadas en fotografía digital y restauración de contenido histórico. La síntesis de voz y generación de música, como el desarrollo de voces artificiales en asistentes virtuales y su utilización de creación de modelos en el sector biomédico, como la generación de imágenes médicas sintéticas para mejorar la precisión en el entrenamiento de modelos de diagnóstico [73,74].

A pesar de sus impresionantes capacidades, las GANs presentan varios desafíos, como la inestabilidad en el entrenamiento, el problema de colapso de modo (cuando el generador solo produce un subconjunto limitado de datos) y la dificultad para evaluar la calidad de los datos generados. Sin embargo, el avance en variantes como Wasserstein GAN (WGAN) ha permitido mejorar la estabilidad del entrenamiento y la diversidad en la generación de datos [73,75].

2.3.2.3 Redes Neuronales Recurrentes (RNN - Recurrent Neural Networks)

Son una clase de redes neuronales especializadas en el procesamiento de secuencias de datos, como series temporales, procesamiento de lenguaje natural y análisis de voz. A diferencia de los modelos de redes neuronales tradicionales, que procesan cada dato de entrada de manera independiente, las RNNs pueden mantener información sobre estados anteriores, lo que les permite capturar relaciones temporales y contextuales en los datos [76].

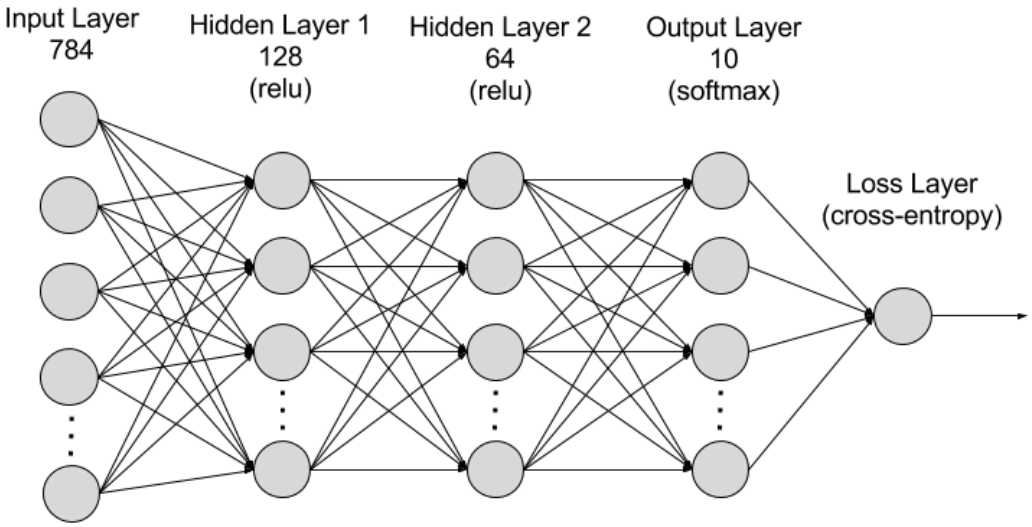


Figura 13.– Arquitectura de una red neuronal recurrentes (RNN).

Red neuronal recurrrente con dos capas ocultas y funciones de activación ReLU, una capa de salida con softmax para clasificación multiclase, y una capa de pérdida de entropía cruzada para el entrenamiento supervisado.

El elemento clave de las RNNs es la existencia de una conexión recurrente en sus neuronas, lo que permite que los estados ocultos sean actualizados en función de la entrada actual y la memoria de pasos previos. Esta propiedad hace que las RNNs sean ideales para tareas como la traducción automática, donde el significado de una palabra puede depender de su contexto anterior, predicción de series temporales, como los modelos utilizados en pronósticos financieros y meteorológicos, reconocimiento de voz, aplicado en asistentes virtuales como Siri, Alexa y Google Assistant así como de generación de texto, usada en chatbots y procesamiento de lenguaje natural [76].

A pesar de sus ventajas, las RNNs presentan el problema de desvanecimiento del gradiente, lo que dificulta la capacidad de la red para capturar dependencias a largo plazo en secuencias extensas. Para abordar esta limitación, se han desarrollado arquitecturas avanzadas como LSTM (Long Short-Term Memory) y GRU (Gated Recurrent Unit) [76,77].

2.3.2.4 LSM (Liquid State Machines)

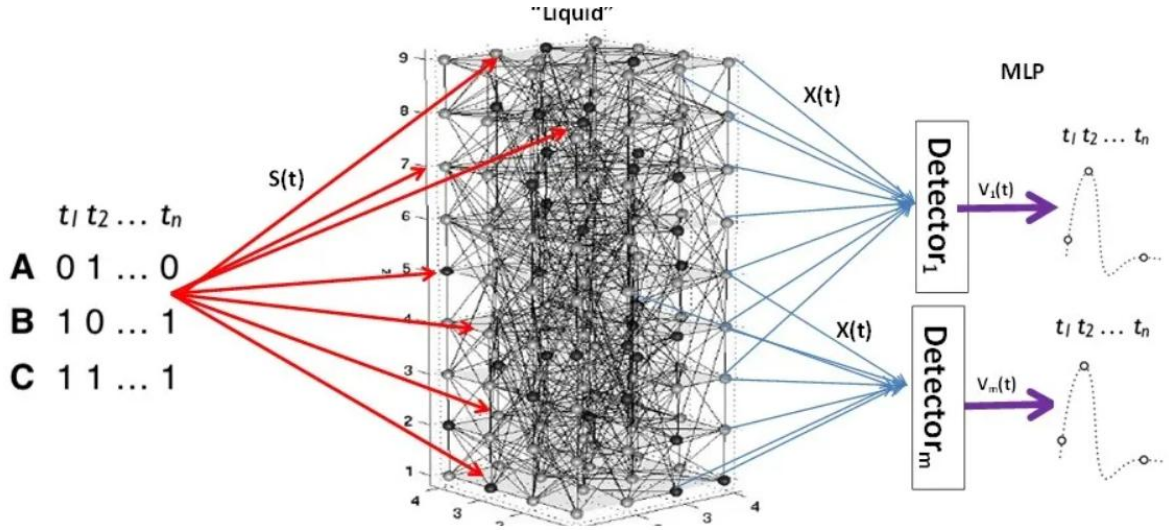


Figura 14.– Representación de una Liquid State Machine (LSM).

Flujo de entrada binaria a una red neuronal recurrente altamente conectada (la "líquida"), cuyo estado dinámico es procesado por detectores lineales o redes MLP para generar salidas temporales. Las LSM son utilizadas para el procesamiento de información en series temporales, simulando mecanismos de memoria y evolución en el tiempo.

Las Liquid State Machines (LSMs) son una variante de las RNNs inspiradas en la neurociencia computacional. A diferencia de los modelos convencionales, las LSMs funcionan como una red de neuronas impulsadas por eventos en la que los datos de entrada son procesados a través de un conjunto dinámico de conexiones neuronales que evolucionan en el tiempo [78,79].

Permite que las LSMs sean altamente eficientes en la simulación de señales neuronales, lo que las hace útiles en aplicaciones como modelado de sistemas biológicos, análisis de actividad cerebral y procesamiento de señales en neurociencia. Su capacidad para representar de manera eficiente información compleja en el tiempo las convierte en una herramienta clave en la investigación de sistemas dinámicos y modelado de la cognición humana [78,79].

2.3.2.5 LSTM (Long Short-Term Memory - Memoria a Largo Corto Plazo)

Las LSTM son una de las arquitecturas más utilizadas para procesamiento de secuencias y dependencias a largo plazo. Fueron desarrolladas para solucionar los problemas de desvanecimiento y explosión del gradiente que afectan a las RNNs convencionales. El componente central de una LSTM es su estructura de celdas de memoria, las cuales incluyen una puerta de entrada que determina qué información de la entrada actual se almacena en la celda de memoria, una puerta de olvido que decide qué parte de la memoria pasada debe ser descartada y una puerta de salida: Regula qué información se envía a la siguiente neurona [80].

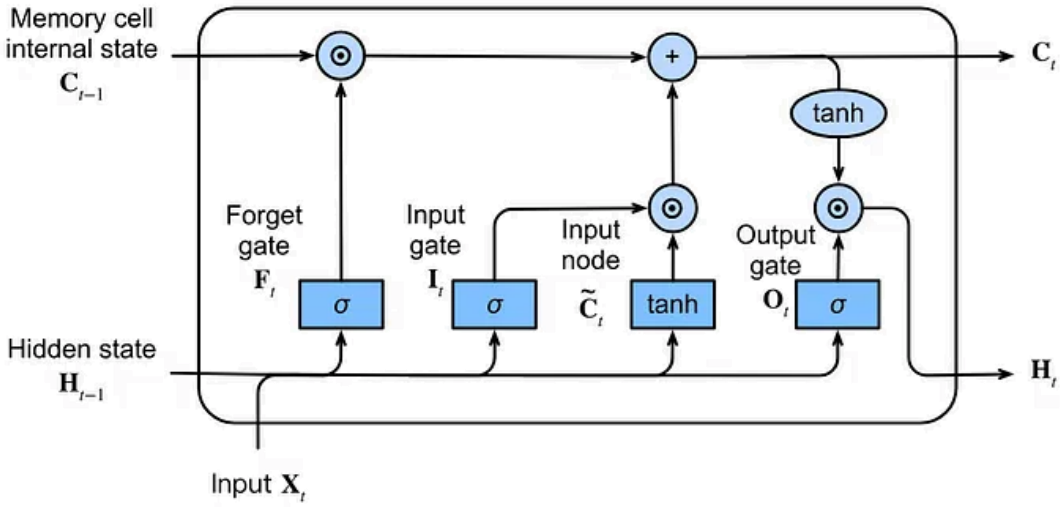


Figura 15.– Arquitectura interna de una celda LSTM.

Estructura de una Long Short-Term Memory (LSTM), incluyendo sus compuertas de olvido, entrada y salida, que permiten regular el flujo de información y preservar dependencias a largo plazo en secuencias temporales.

Las LSTM son ampliamente utilizadas en la traducción automática (Google Translate), Modelado del lenguaje (GPT-4 y otros modelos de lenguaje), predicción de valores en series temporales (mercados financieros, pronósticos de demanda) y análisis de texto y sentimiento, como en redes sociales o detección de spam [80,81].

2.3.2.6 GRU (Gated Recurrent Unit - Unidad Recurrente Gated)

Las GRUs son una variante simplificada de las LSTM que mantienen un rendimiento similar, pero con una estructura menos compleja y más eficiente en términos computacionales. A diferencia de las LSTM, las GRUs combinan las puertas de entrada y olvido en una sola puerta, lo que reduce la cantidad de parámetros a entrenar. Esto hace que las GRUs sean más rápidas de entrenar y menos propensas al sobreajuste, especialmente cuando se trabaja con conjuntos de datos limitados. Su principal uso es en el reconocimiento de voz en asistentes inteligentes, análisis de series temporales en biomedicina y economía y en la generación de texto en modelos como chatbots [82].

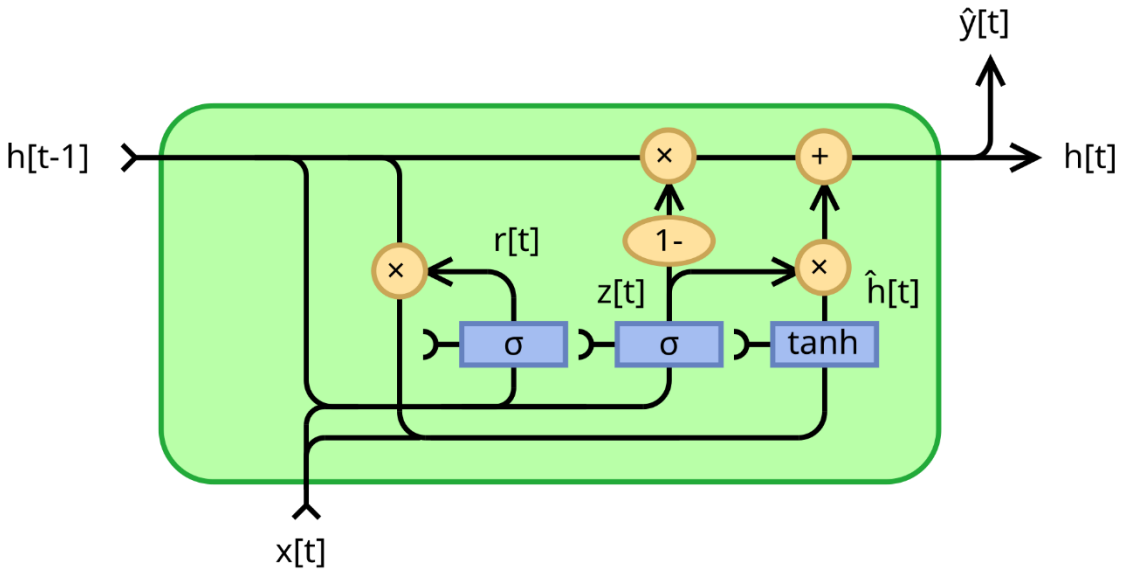


Figura 16.– Arquitectura interna de una unidad GRU (Gated Recurrent Unit).

Funcionamiento de una GRU, una variante de red neuronal recurrente que utiliza compuertas de actualización (z[t]) y reinicio (r[t]) para controlar el flujo de información, permitiendo capturar dependencias temporales sin necesidad de una celda de memoria explícita como en las LSTM.

2.3.2.7 Redes Neuronales Convolucionales (CNN - Convolutional Neural Networks)

Las CNNs son un tipo de red neuronal especializada en procesamiento de imágenes y señales espaciales. Su arquitectura está inspirada en el sistema visual biológico y se basa en capas convolucionales que extraen características clave de los datos de entrada. Las principales aplicaciones de las CNNs se utilizan en diagnóstico médico mediante análisis

de imágenes de resonancia magnética y radiografías; reconocimiento facial en aplicaciones de seguridad y dispositivos móviles y detección de objetos en conducción autónoma y sistemas de vigilancia [83].

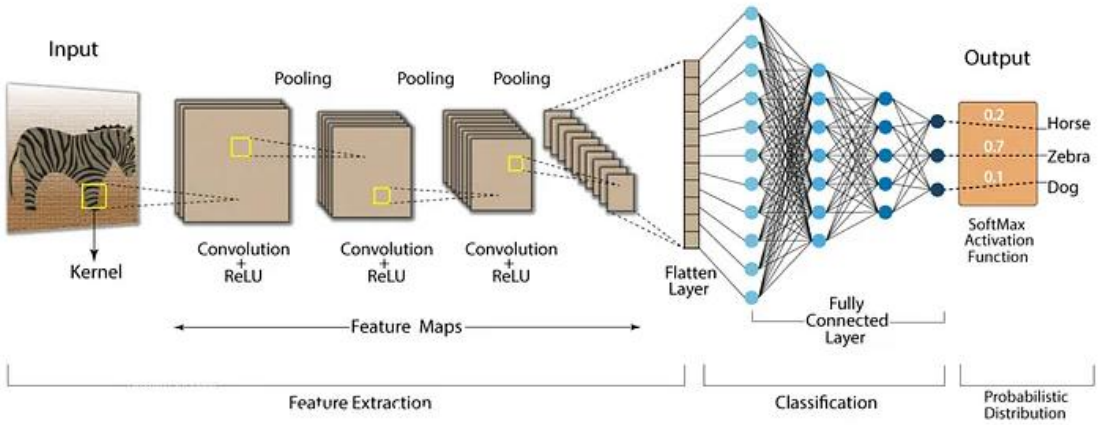


Figura 17.– Arquitectura de una Red Neuronal convolucional (CNN)

Funcionamiento de una CNN, donde una imagen de entrada atraviesa múltiples capas de convolución, activación y agrupamiento (pooling) para extraer características, que luego se procesan mediante capas densas para clasificar la imagen en una distribución probabilística de categorías. Tomado de https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5

2.3.2.8 DCNN (Deep Convolutional Neural Networks - Red Neuronal Convolucional Profunda)

Las DCNNs son una extensión de las CNNs que incorporan capas adicionales para mejorar la capacidad de generalización y detección de patrones en datos de alta complejidad. Su uso ha sido clave en el desarrollo de redes neuronales profundas como ResNet, VGGNet y EfficientNet, aplicadas en reconocimiento de imágenes, biometría y diagnóstico médico. Las DCNNs han sido particularmente útiles en el diagnóstico de enfermedades mediante el análisis automático de imágenes médicas, ayudando a detectar patologías como cáncer de piel, neumonía y anomalías en exámenes de retina con niveles de precisión comparables a los de un especialista humano [84].

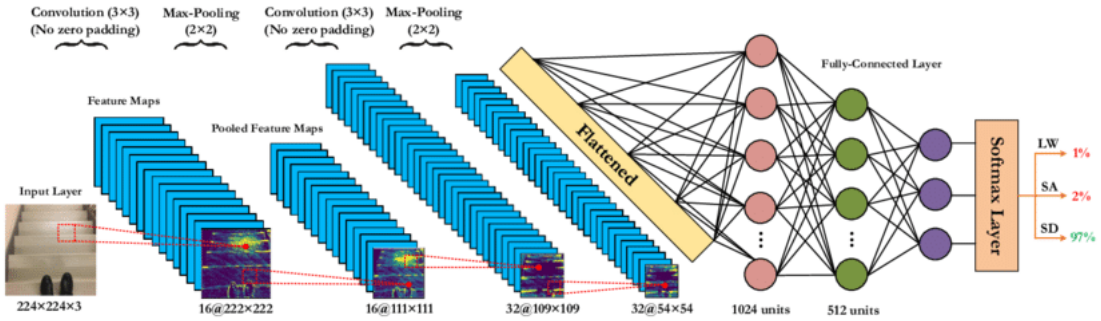


Figura 18.– Flujo de procesamiento en una Red Neuronal Convolucional Profunda (DCNN).

Etapas de una DCNN típica: desde la imagen de entrada, pasando por capas de convolución y max-pooling para extraer características, hasta la capa densa totalmente conectada y la capa softmax que genera la probabilidad de clasificación en distintas categorías. Tomado de https://www.researchgate.net/figure/ARCHITECTURE-OF-A-DEEP-CONVOLUTIONAL-NEURAL-NETWORK-WITH-TWO-CONVOLUTION-LAYERS-WITH-16_fig3_333882146

2.4 Aprendizaje por Refuerzo (Reinforcement Learning)

El aprendizaje por refuerzo (Reinforcement Learning, RL) es un paradigma dentro del aprendizaje automático en el que un agente aprende a tomar decisiones mediante un proceso de prueba y error en un entorno dinámico. A diferencia del aprendizaje supervisado, donde un modelo aprende a partir de ejemplos etiquetados, el aprendizaje por refuerzo se basa en la interacción con el entorno, en el que el agente recibe recompensas o penalizaciones dependiendo de sus acciones [85].

Este enfoque está inspirado en la teoría del condicionamiento operante en psicología, donde un sistema aprende comportamientos óptimos a partir de la retroalimentación obtenida de su entorno. El objetivo principal de RL es maximizar la recompensa acumulada a lo largo del tiempo, optimizando la estrategia del agente a través de múltiples interacciones con el entorno [86,87].

El aprendizaje por refuerzo ha sido aplicado con éxito en robots autónomos, videojuegos, finanzas, control de sistemas, automatización industrial y diagnóstico médico. Sus avances más recientes han permitido desarrollar modelos capaces de jugar videojuegos con un nivel superior al humano (como AlphaGo de DeepMind), mejorar el control de tráfico en ciudades inteligentes y diseñar fármacos mediante simulaciones químicas avanzadas [87,88].

El marco matemático del aprendizaje por refuerzo se basa en los Procesos de Decisión de Markov (MDP), donde el agente aprende una política óptima $\pi(s)$ que le permite seleccionar la mejor acción aa para cada estado ss, con el fin de maximizar su recompensa esperada R_t [87–89].

2.4.1 Algoritmos Genéticos

Los algoritmos genéticos (Genetic Algorithms, GA) son una técnica de optimización inspirada en la evolución biológica. Estos algoritmos buscan resolver problemas complejos mediante la simulación de los principios de la selección natural, tales como mutación, cruce y supervivencia del más apto [88,90].

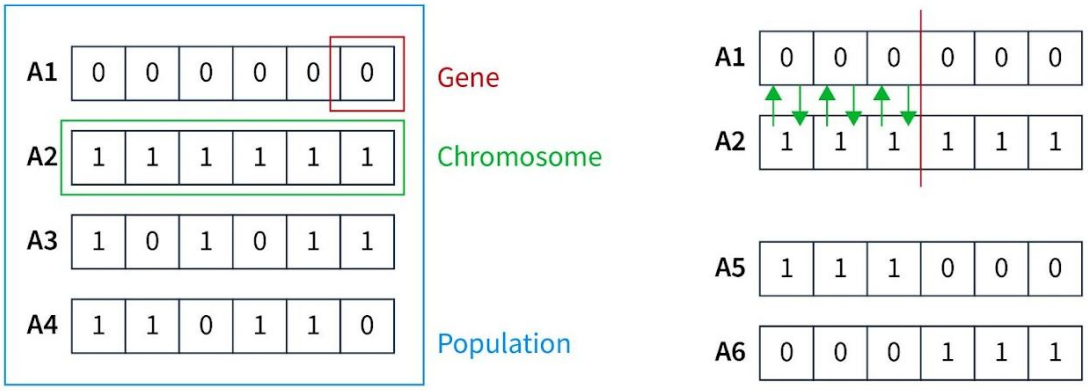


Figura 19.– Representación de un algoritmo genético.

Componentes clave de un algoritmo genético: una población de soluciones codificadas como cromosomas (filas de bits), cada uno compuesto por genes, y el proceso de cruces (crossover) entre cromosomas para generar nuevas soluciones en la siguiente generación. Tomado de https://www.appliedaicourse.com/blog/genetic-algorithm-in-machine-learning/

En un algoritmo genético, una población de soluciones candidatas evoluciona iterativamente hacia una solución óptima. El proceso se lleva a cabo en varias generaciones, donde las soluciones más exitosas (aquellas que maximizan una función de aptitud) se combinan y modifican para producir una nueva población con características mejoradas [88,90].

Las etapas principales de un algoritmo genético, funcionan como un proceso evolutivo, se da una inicialización, que genera una población aleatoria de soluciones, posteriormente se evalua la misma, es decir cada individuo es evaluado según una función de aptitud, se hace una selección, eligiendo los mejores individuos para la reproducción, se realiza un cruzamiento (Crossover), para combinar las soluciones seleccionadas para crear nuevas, se da un impulso evolutivo, es decir se hace una mutación, introduciendo pequeñas variaciones para evitar convergencias prematuras y se reemplaza, la población existente por la nueva población con los individuos generados producto del crossover [90].

2.4.2 Q-Learning

El Q-Learning es un algoritmo de aprendizaje por refuerzo basado en valores que permite a un agente aprender la mejor política de acción en entornos desconocidos. Funciona almacenando en una tabla $Q(s,a)$ las recompensas esperadas para cada par estado-acción, las cuales son actualizadas conforme el agente explora el entorno [91]. El aprendizaje en Q-Learning sigue la ecuación de actualización de Bellman:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a))$$

Ecuación 1.– ecuación de actualización de Bellman

Donde:

α es la tasa de aprendizaje.

γes el factor de descuento, que pondera la importancia de recompensas futuras.

r es la recompensa obtenida por tomar la acción a en el estado s.

$max_{a'}\, Q(s',a')$ representa la mejor acción futura esperada.

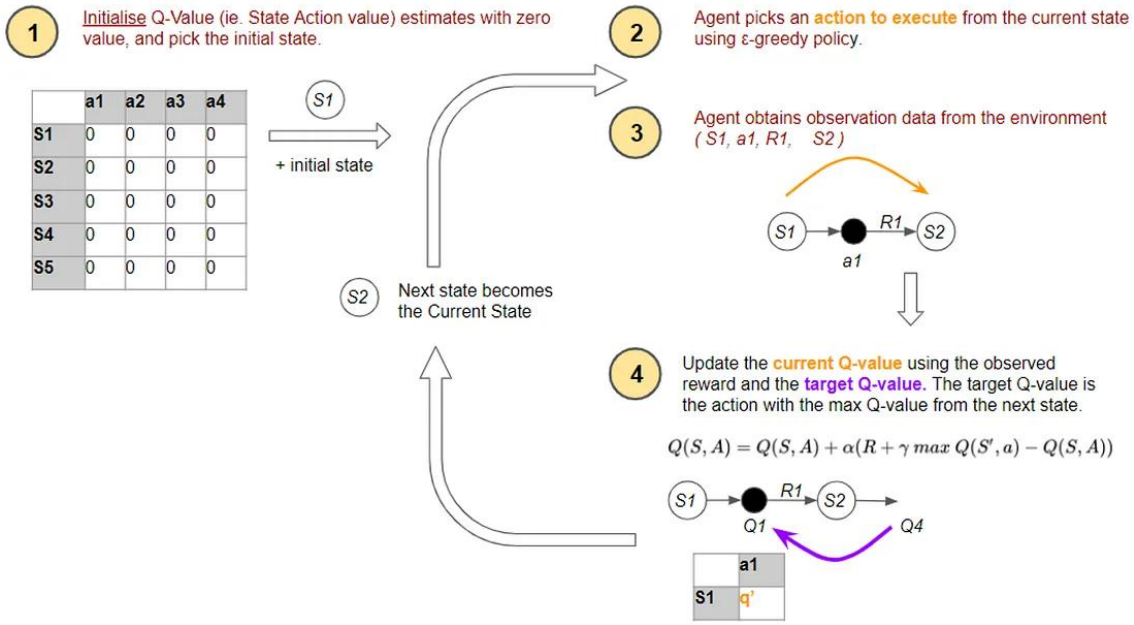


Figura 20.– Ciclo de actualización del algoritmo Q-Learning.

La imagen describe el proceso iterativo de aprendizaje por refuerzo basado en Q-Learning: inicialización de valores Q, selección de acción mediante política ε-greedy, observación del nuevo estado y recompensa, y actualización del valor Q usando la recompensa recibida y la mejor estimación futura. Tomado de https://medium.com/data-science/reinforcement-learning-explained-visually-part-4-q-learning-step-by-step-b65efb731d3e

Una de las principales limitaciones del Q-Learning es la necesidad de mantener una tabla Q que crece exponencialmente con el número de estados y acciones. Para abordar este problema, se han desarrollado versiones mejoradas como Deep Q-Network (DQN) [92].

2.4.2.1 Deep Q-Network (DQN)

El Deep Q-Network (DQN) es una extensión del Q-Learning que incorpora redes neuronales profundas para aproximar la función de valor $Q(s, a)$, eliminando la necesidad de almacenar una tabla QQQ explícita, lo que permite su aplicación en entornos de alta dimensionalidad. Desarrollado por DeepMind, este enfoque ha sido fundamental en la evolución del aprendizaje por refuerzo profundo, posibilitando la toma de decisiones en entornos complejos donde el número de estados y acciones es demasiado grande para ser manejado por métodos tradicionales [92].

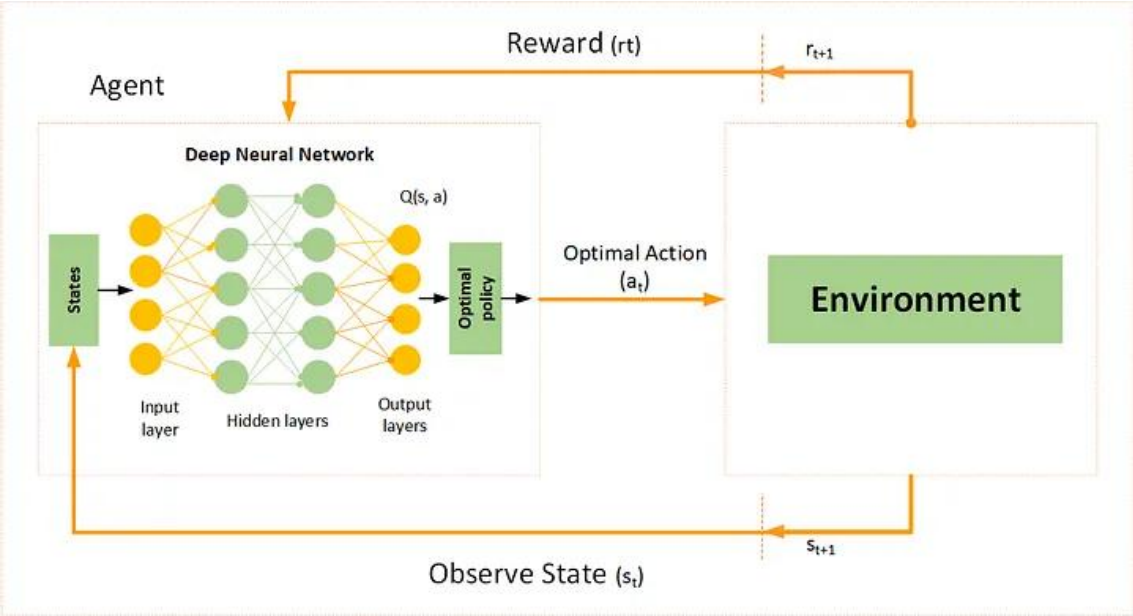


Figura 21.– Esquema de Deep Q-Learning (DQN).

Interacción entre un agente y el entorno mediante aprendizaje por refuerzo profundo. Una red neuronal estima los valores Q para cada acción posible dado un estado, lo que permite seleccionar la política óptima. El agente ejecuta una acción, recibe una recompensa y observa el nuevo estado, cerrando el ciclo de aprendizaje. Tomado de https://medium.com/@samina.amin/deep-q-learning-dqn-71c109586bae

Entre sus principales mejoras, el DQN introduce el mecanismo de memoria de experiencia (Experience Replay), que permite almacenar transiciones pasadas y reutilizarlas en futuras actualizaciones, lo que rompe la correlación temporal de los datos y estabiliza el entrenamiento. Además, emplea una red neuronal objetivo (Target Network) para reducir la variabilidad en la actualización de los valores Q, mejorando la convergencia del modelo [89,92].

2.4.2.2 SARSA (State-Action-Reward-State-Action)

El algoritmo SARSA es una variante de Q-Learning que actualiza sus valores de estado-acción considerando la acción que el agente realmente tomará en el siguiente paso, en lugar de basarse en la acción óptima.

La ecuación de actualización de SARSA es:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

Ecuación 2.– ecuación de actualización de SARSA

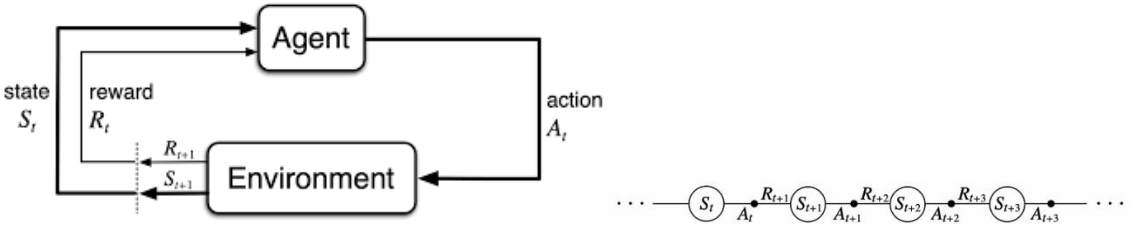


Figura 22.– Ciclo de interacción agente-entorno y Cadena de decisiones en aprendizaje por refuerzo – SARSA

Bucle fundamental del aprendizaje por refuerzo, donde un agente observa un estado S_t, ejecuta una acción A_t, y recibe del entorno una nueva observación S_{t+1} y una recompensa R_{t+1}, ajustando su comportamiento en función de esta retroalimentación y secuencia temporal de estados S_t, acciones A_t y recompensas R_{t+1}, donde cada decisión afecta el siguiente estado y recompensa.

La diferencia clave con Q-Learning radica en que este último usa la mejor acción posible en el futuro, mientras que SARSA utiliza la acción que realmente tomará el agente según su política actual [89,93].

2.4.2.3 A3C (Asynchronous Advantage Actor-Critic)

El A3C (Asynchronous Advantage Actor-Critic) combina de manera efectiva los enfoques de aprendizaje basado en políticas y en valores, lo que permite mejorar significativamente la eficiencia del entrenamiento en entornos complejos y de alta dimensionalidad. Desarrollado por DeepMind, este modelo representa un avance respecto a métodos anteriores como Q-Learning y Deep Q-Networks (DQN), al introducir una estrategia de aprendizaje asincrónico que optimiza la exploración y estabilidad del aprendizaje [89,94].

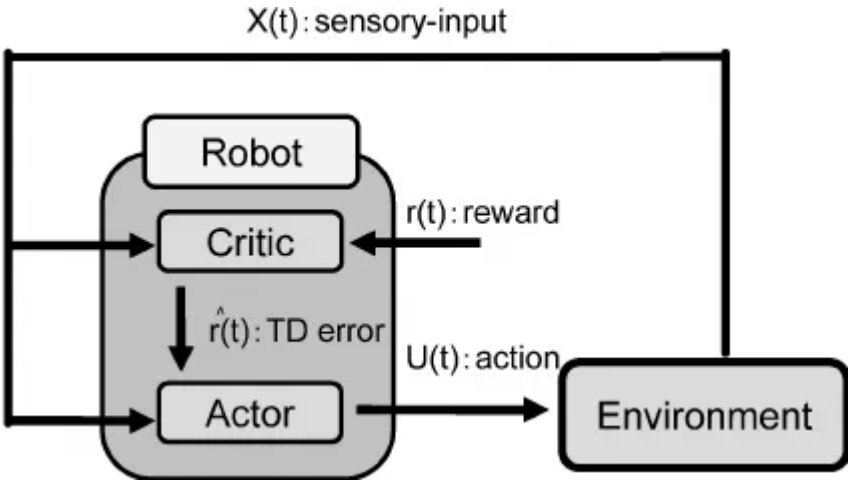


Figura 23.– Esquema del modelo Actor-Critic en aprendizaje por refuerzo.

Arquitectura Actor-Critic, donde el actor toma decisiones (acciones) basadas en el estado actual del entorno, mientras que el crítico evalúa esas acciones generando una señal de error temporal (TD error), que sirve como retroalimentación para mejorar la política del agente.

Una de las características más relevantes del A3C es su capacidad para ejecutar múltiples agentes en paralelo, lo que reduce la correlación entre muestras y mejora la convergencia del modelo, evitando problemas como el sobreajuste a trayectorias específicas del agente. Además, este algoritmo se basa en la arquitectura Actor-Critic, en la cual el Actor aprende una política óptima para la selección de acciones, mientras que el Critic estima el valor de los estados y proporciona una retroalimentación ajustada para mejorar la toma de decisiones. Este enfoque permite un balance entre la exploración de nuevas estrategias y la explotación de aquellas que han demostrado ser efectivas, optimizando la búsqueda de una política óptima en entornos dinámicos [89,94].

2.5 Evolución de los Modelos de Lenguaje Grande (LLM)

Los Modelos de Lenguaje Grande (LLMs, por sus siglas en inglés: Large Language Models) representan un avance significativo en el campo del Procesamiento del Lenguaje Natural (NLP) y la inteligencia artificial. Se trata de modelos basados en redes neuronales profundas que han sido entrenados con enormes volúmenes de datos textuales para desarrollar una comprensión avanzada del lenguaje humano. Su capacidad para generar texto, responder preguntas, traducir idiomas y realizar tareas lingüísticas complejas ha

impulsado su adopción en diversos campos, desde la automatización de procesos hasta la investigación científica [95,96].

Los LLMs funcionan mediante el uso de técnicas de aprendizaje profundo, en particular, mediante arquitecturas de redes neuronales basadas en Transformers, las cuales han demostrado una capacidad sin precedentes para modelar patrones lingüísticos. A través de un proceso de entrenamiento supervisado y no supervisado, los LLMs pueden predecir la siguiente palabra en una secuencia de texto con base en el contexto previo, lo que les permite generar contenido con un alto grado de coherencia y precisión [95].

Estos modelos han evolucionado desde arquitecturas tradicionales como los modelos n-gram, Word2Vec y GloVe, hasta el desarrollo de transformers como GPT-4, BERT, T5 y PaLM, los cuales han demostrado una mejora sustancial en tareas como resumen automático de textos, generación de código, análisis de sentimientos, asistencia en la toma de decisiones médicas y automatización del análisis estadístico. Su impacto en la inteligencia artificial ha sido tal, que actualmente constituyen el núcleo de muchos agentes conversacionales y sistemas de soporte basados en IA [40,95,97].

A pesar de sus capacidades avanzadas, los LLMs presentan desafíos importantes, incluyendo la necesidad de enormes recursos computacionales, posibles sesgos en los datos de entrenamiento, dificultades en la interpretabilidad de sus decisiones, y el riesgo de generar respuestas erróneas o inexactas. Para abordar estas limitaciones, se han desarrollado estrategias como ajustes finos (fine-tuning), ingeniería de prompts (prompt engineering) y técnicas de aprendizaje por refuerzo con retroalimentación humana (RLHF, Reinforcement Learning from Human Feedback), con el fin de mejorar su precisión y aplicabilidad en escenarios específicos [42,95].

2.5.1 Transformers: La Arquitectura Base de los Modelos de Lenguaje Grande

Los Transformers son un tipo de red neuronal profunda que ha revolucionado el campo del Procesamiento del Lenguaje Natural (NLP) y el aprendizaje profundo en general.

Introducidos por Vaswani et al. (2017) en el artículo "Attention is All You Need" [98], los Transformers han demostrado una capacidad superior en el procesamiento de secuencias de texto, reemplazando modelos previos como las Redes Neuronales Recurrentes (RNNs), Long Short-Term Memory (LSTM) y Gated Recurrent Units (GRU) [99–101].

La clave del éxito de los Transformers radica en su mecanismo de autoatención (self-attention), el cual permite al modelo analizar todas las palabras de un texto simultáneamente, en lugar de procesarlas secuencialmente como lo hacen las RNNs. Esto le confiere una capacidad superior para capturar relaciones de largo alcance entre palabras y mejorar la coherencia contextual en la generación de texto [101,102].

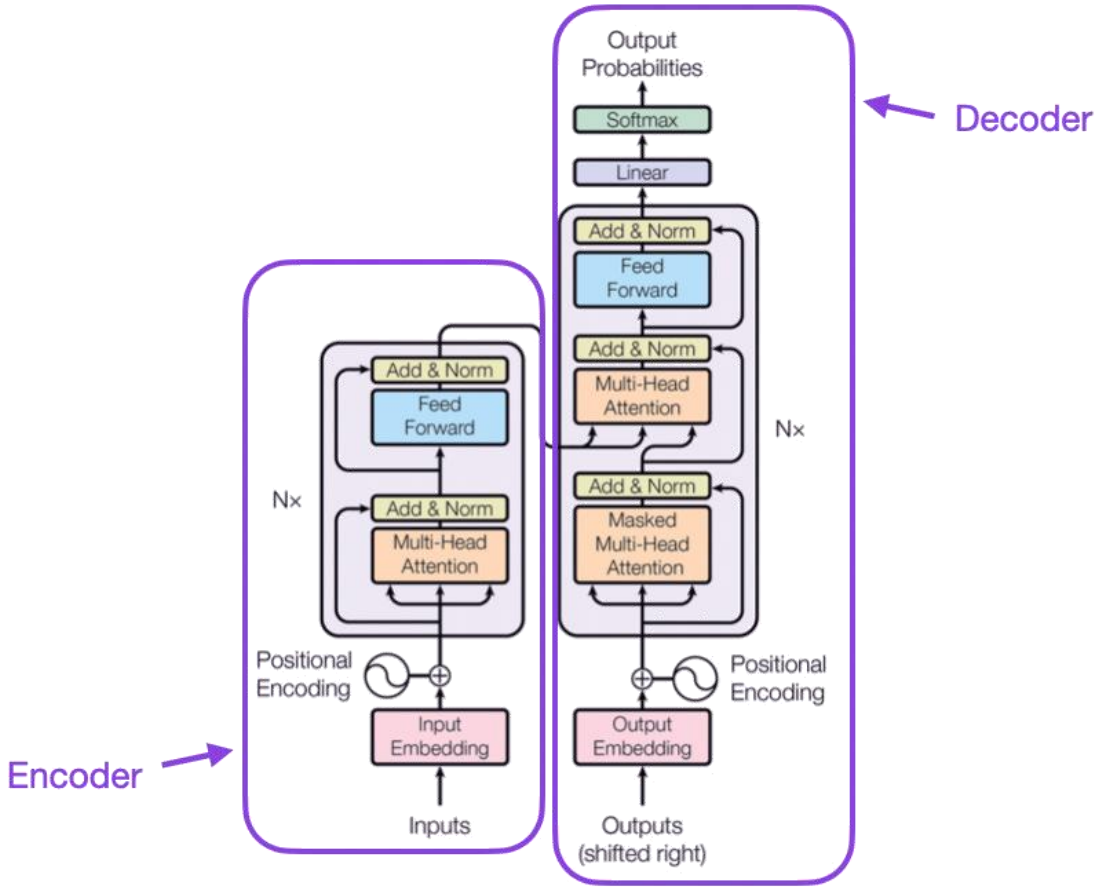


Figura 24.– Arquitectura del modelo Transformer.

Modelo Transformer, compuesto por un codificador (encoder) y un decodificador (decoder), ambos formados por bloques repetidos de atención multi-cabeza, normalización y capas feed-forward. Este diseño permite capturar dependencias a largo plazo en secuencias, y es la base de los Modelos de Lenguaje Grande (LLMs). Tomado de: https://arxiv.org/abs/1706.03762

El mecanismo de autoatención se basa en el cálculo de pesos de atención para cada palabra en una secuencia, lo que permite que el modelo priorice la información relevante y reduzca la influencia de palabras irrelevantes en la comprensión del contexto. Este enfoque ha mejorado significativamente el rendimiento de los modelos de NLP en tareas como traducción automática, generación de texto, clasificación de documentos y análisis de sentimientos [101,102].

Los Transformers están compuestos por capas de autoatención multi-cabeza y redes feedforward, lo que les permite procesar texto con una escalabilidad y paralelización superiores a las RNNs. Gracias a esto, los modelos basados en Transformers pueden ser entrenados en grandes cantidades de datos utilizando hardware especializado como GPU (Graphics Processing Units) y TPU (Tensor Processing Unit), lo que ha impulsado el desarrollo de Modelos de Lenguaje Grande (LLMs) cada vez más potentes [101,102].

La evolución de los Transformers ha dado lugar a diversas arquitecturas especializadas, como BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), T5 (Text-to-Text Transfer Transformer) y PaLM (Pathways Language Model) [103,104], cada una de ellas optimizada para tareas específicas dentro del NLP. Estos modelos han sido implementados en aplicaciones que van desde asistentes virtuales y chatbots hasta sistemas de diagnóstico médico y generación automatizada de código [99–101].

2.5.2 LLMs y Agentes Inteligentes

El desarrollo de Modelos de Lenguaje Grande (LLMs) ha trascendido su función original de interpretar y generar texto, evolucionando hacia sistemas de inteligencia artificial más complejos que integran capacidades de procesamiento del lenguaje natural con toma de decisiones autónomas. Esta transformación ha dado lugar a la creación de agentes inteligentes basados en lenguaje, los cuales pueden interactuar con entornos dinámicos, interpretar información en tiempo real y ejecutar tareas automatizadas con un nivel de autonomía sin precedentes [105].

Figura 25.– Estructura funcional de un agente inteligente basado en LLM.

La imagen muestra cómo interactúan los componentes de un agente IA: el usuario envía instrucciones que el LLM interpreta, divide en subtareas (planificación), consulta memoria o documentos (RAG), acciona herramientas externas, y finalmente entrega una solución ajustada por retroalimentación. Tomado de https://itcl.es/blog/que-son-y-como-funcionan-los-chatbots-y-los-agentes-ia/

Los agentes inteligentes son sistemas autónomos diseñados para percibir su entorno, procesar información y ejecutar acciones con el fin de lograr objetivos específicos. En el contexto de la inteligencia artificial (IA), un agente inteligente es una entidad computacional capaz de tomar decisiones de manera autónoma y optimizar su comportamiento mediante el aprendizaje y la adaptación continua [106]. Estos agentes han sido ampliamente utilizados en automatización de procesos, robótica, sistemas de recomendación, modelado de entornos dinámicos y procesamiento del lenguaje natural [44].

El desarrollo de agentes inteligentes ha sido impulsado por avances en aprendizaje profundo, aprendizaje por refuerzo y modelos de lenguaje grande (LLMs), lo que ha permitido la creación de sistemas capaces de razonar, interactuar con humanos y mejorar su desempeño a lo largo del tiempo. Dependiendo de su diseño y nivel de autonomía, estos agentes pueden operar en entornos controlados o en sistemas abiertos y complejos, donde deben adaptarse a condiciones cambiantes e incertidumbre en la toma de decisiones [42].

2.5.2.1 Características y Funcionamiento Fundamental de un Agente

Para ser considerado «inteligente», un agente debe poseer ciertas características esenciales que le permitan percibir, procesar información y actuar en consecuencia. Un agente debe tener la capacidad para operar sin intervención humana directa, tomando decisiones en función de la información disponible, en algunos casos debe poder tener una percepción del entorno físicos (en el caso de robots) o virtuales (como APIs y bases de datos) para recopilar información relevante. También deben tener cierta capacidad de razonamiento con el cual puedan evaluar los datos y seleccionar las acciones pertinentes y que están basadas en modelos de decisión, reglas lógicas o aprendizaje automático. Y por último debe tener capacidad de adaptabilidad y aprendizaje, lo que le da la posibilidad de mejorar su desempeño a partir de experiencias pasadas, ajustando su comportamiento en función de la retroalimentación recibida [33,36].

Otros dos aspectos que debe tener un agente es que debe poder tener interacción con otros sistemas, humanos, otros agentes o software especializado para optimizar tareas colaborativas. Debe siempre buscar la optimización de objetivos, mediante la priorización de acciones en función de un criterio de desempeño predefinido, como maximizar la eficiencia o minimizar errores [33,36].

Los agentes inteligentes pueden ser diseñados para tareas específicas, como los sistemas de detección de fraudes, asistentes virtuales y diagnóstico médico automatizado, o pueden ser más generales, como los agentes conversacionales avanzados impulsados por Modelos de Lenguaje Grande (LLMs) [39,44].

2.5.2.2 Clasificación de los Agentes Inteligentes

Los agentes inteligentes pueden clasificarse en diferentes categorías según su nivel de autonomía, capacidad de adaptación y ámbito de aplicación. Están los Agentes Reactivos, los cuales responden a estímulos del entorno sin la capacidad de planificar o almacenar memoria, como lo son los sensores en sistemas de control industrial o asistentes virtuales básicos. También están los Agentes Basados en Modelo, que mantienen una

representación interna del entorno y pueden tomar decisiones informadas, los sistemas de diagnóstico médico basados en reglas, son ejemplo de ello [107]. Los Agentes Basados en Objetivos, este tipo de agentes optimizan su comportamiento para alcanzar metas específicas, un claro ejemplo de ellos son los sistemas de optimización logística o motores de recomendación en comercio electrónico. Están los Agentes Basados en Aprendizaje, estos aprenden de la experiencia y ajustan su comportamiento dinámicamente, el trading financiero utiliza este tipo de agentes. Y por último los Agentes Multiagente, que como su nombre lo indica, estos combinan múltiples agentes que interactúan entre sí para resolver tareas colaborativas, los sistemas regulatorios de tráfico en ciudades modernas [108].

2.5.2.3 Interacción entre los Modelos de Lenguaje Grande (LLMs) y los Agentes Inteligentes

Los LLMs han evolucionado hasta convertirse en el núcleo cognitivo de numerosos agentes de IA, ya que su capacidad para procesar, comprender y generar lenguaje natural les permite interactuar con diversos sistemas informáticos, bases de datos y herramientas analíticas. Estos modelos han sido diseñados para interpretar instrucciones en lenguaje natural y traducirlas en acciones concretas dentro de entornos digitales, lo que reduce significativamente la necesidad de interfaces programáticas tradicionales y permite la automatización de procesos en una amplia variedad de aplicaciones científicas y tecnológicas [44].

En la investigación biomédica y el análisis estadístico automatizado, los LLMs han demostrado ser herramientas clave para la optimización de flujos de trabajo. Su integración con software estadístico y entornos de programación permite la generación automatizada de código en lenguajes como Python y R, facilitando la ejecución de modelos predictivos, análisis de datos y generación de reportes científicos detallados [43]. Además, estos modelos pueden proporcionar explicaciones en lenguaje natural sobre los resultados obtenidos, mejorando la accesibilidad de técnicas avanzadas para investigadores sin formación especializada en bioestadística. Esta capacidad de los LLMs para transformar el análisis de datos en información comprensible no solo optimiza la toma de decisiones, sino que también fomenta la transparencia y reproducibilidad de los estudios científicos.

La efectividad de los agentes inteligentes basados en LLMs radica en su capacidad de integración con otros sistemas avanzados de inteligencia artificial, en particular con distintos tipos de redes neuronales diseñadas para abordar problemas específicos. Dependiendo de la naturaleza de la tarea, los LLMs pueden interactuar con diferentes arquitecturas neuronales, ampliando así su aplicabilidad en distintos dominios [44].

Al combinar CNNs con los LLMs para el análisis de imágenes médicas, se puede llegar a interpretar una imagen de rayos X y un LLM para generar un informe automático en lenguaje natural sobre los hallazgos detectados. Con las RNNs pueden emplearse junto con LLMs en la predicción de tendencias estadísticas, el análisis de series temporales en epidemiología o la detección de patrones en grandes volúmenes de datos clínicos. Y con los RL los agentes pueden llegar a operar en entornos dinámicos y optimizar la automatización de procesos complejos, como la gestión de experimentos en laboratorios biomédicos o la toma de decisiones en sistemas de salud basados en IA [32,33].

2.5.3 RAG y Agentes Inteligentes Basados en LLMs: Sinergia para el Análisis Automatizado

El paradigma Retrieval-Augmented Generation (RAG) ha transformado el uso de Modelos de Lenguaje Grande (LLMs) al combinar sus capacidades generativas con mecanismos eficientes de recuperación de información desde bases vectoriales externas. La generación aumentada por recuperación (Retrieval-Augmented Generation, RAG) es una arquitectura híbrida que combina modelos generativos, como los LLMs, con mecanismos de recuperación semántica de información externa. Su objetivo principal es mejorar la precisión y relevancia contextual de las respuestas generadas, superando así las limitaciones del conocimiento estático preentrenado en los modelos de lenguaje [36]. El proceso comienza con la generación de un *embedding* semántico de la consulta, que se utiliza como vector de búsqueda dentro de una base de datos vectorial, generalmente implementada con FAISS (Facebook AI Similarity Search) [109].

Esta base documental está compuesta por fragmentos de texto o chunks previamente extraídos, segmentados y embebidos mediante modelos como `text-embedding-3-large`. La segmentación del texto en fragmentos (*chunking*) se realiza comúnmente con solapamiento de tokens para preservar la coherencia contextual y evitar que los límites entre fragmentos interrumpan unidades semánticas clave. Esta técnica permite que los modelos de recuperación mantengan continuidad temática entre fragmentos contiguos, mejorando así la relevancia de los resultados durante la búsqueda por similitud semántica [110,111]. FAISS permite realizar búsquedas por similitud, ya sea por distancia euclidiana (IndexFlatL2) o mediante estructuras más avanzadas como IndexIVFFlat o HNSW, dependiendo del balance entre precisión y velocidad requerido [109,110].

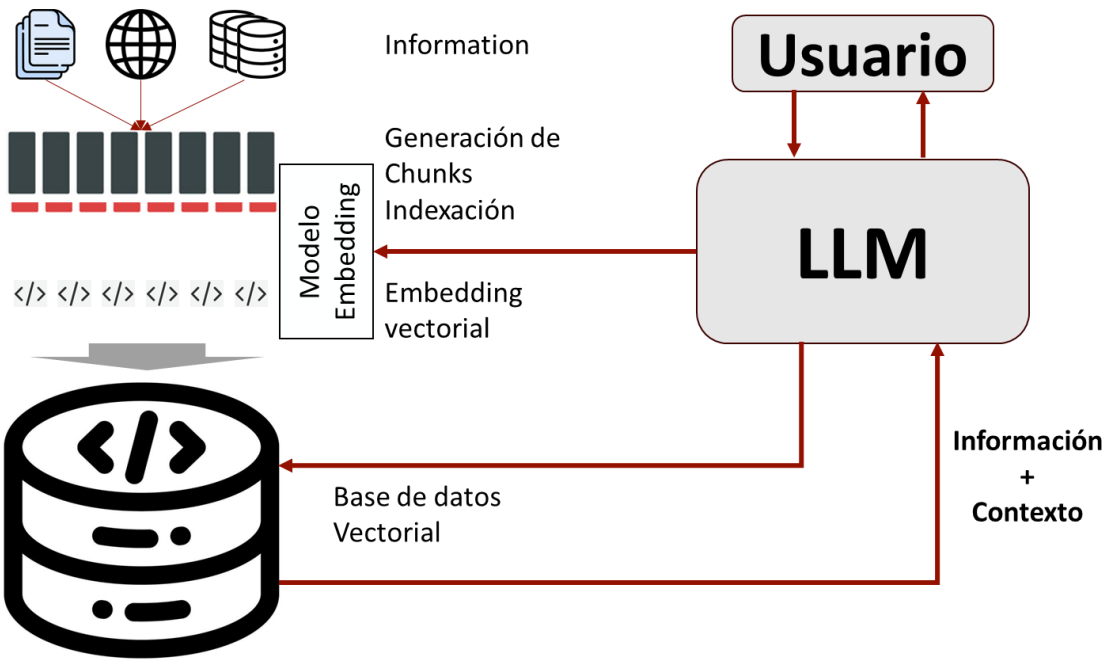


Figura 26.– Arquitectura del enfoque RAG (Retrieval-Augmented Generation) con modelo de lenguaje grande (LLM)

Los documentos se fragmentan con solapamiento y se vectorizan para su almacenamiento en una base de datos semántica. El LLM recupera los fragmentos relevantes a partir de una consulta del usuario y los utiliza como contexto para generar una respuesta más precisa y fundamentada.

Los fragmentos más relevantes recuperados se incorporan al contexto del modelo generador, permitiendo que las respuestas estén fundamentadas en información específica, reciente y trazable. Esto reduce de forma significativa las llamadas

"alucinaciones", es decir, la generación de información plausible pero incorrecta, y fortalece la aplicabilidad del sistema en dominios especializados como la investigación biomédica [25][2,4]. Estudios recientes han confirmado la eficacia de RAG en la generación de datasets clínicos sintéticos [112] y en la mejora de la precisión de LLMs aplicados a tareas científicas [97].

La utilidad de RAG ha sido validada en diversos entornos biomédicos. Barr et al. [112] demostraron que GPT-4o, al emplear RAG, pudo generar datasets sintéticos con propiedades estadísticas comparables a bases clínicas reales como VitalDB, logrando una coincidencia estadística superior al 90%. Por su parte, Lobentanzer et al. [97] destacaron que los LLMs mejoran su rendimiento en tareas de extracción de información biomédica al integrarse con mecanismos de recuperación semántica, permitiendo mayor fidelidad en la interpretación de artículos científicos y guías clínicas.

En arquitecturas multiagente, la recuperación aumentada es clave para garantizar la coherencia y progresividad del razonamiento. El Agente Planificador descompone el problema en subtareas jerárquicas; el Agente Analizador identifica ambigüedades y necesidades de conocimiento externo; y el Agente Investigador ejecuta el componente RAG para proporcionar la información estructurada. Esta interacción orquestada es supervisada por el usuario en un enfoque Human-in-the-Loop, promoviendo control, transparencia y adaptabilidad [113].

Estudios como el de Singhal et al. [25] respaldan que los LLMs equipados con RAG pueden replicar procedimientos y razonamientos clínicos con alta fidelidad, lo cual los posiciona como herramientas valiosas para validar hipótesis, generar código, interpretar resultados estadísticos y producir reportes científicos de forma automatizada y reproducible.

2.5.4 Aplicaciones de los LLMs y Agentes en la Investigación Biomédica

Los agentes han transformado significativamente múltiples sectores, incluyendo la investigación biomédica, la automatización del análisis estadístico y la validación de

estudios científicos. Su integración con LLMs y técnicas avanzadas de aprendizaje profundo ha permitido mejorar la eficiencia y precisión en el manejo de grandes volúmenes de datos, optimizando los procesos de investigación y reduciendo la carga de trabajo manual en tareas complejas [95,114].

Una de las aplicaciones más destacadas de estos agentes es la automatización del análisis estadístico, en la cual los sistemas de IA pueden generar código en lenguajes como R o Python, facilitando la ejecución de pruebas estadísticas y modelos predictivos sin intervención manual. Esta automatización no solo acelera el procesamiento de datos, sino que también disminuye la probabilidad de errores humanos, asegurando análisis más rigurosos y reproducibles [4]. Además, los agentes inteligentes han demostrado su capacidad para optimizar la reproducibilidad científica, al detectar inconsistencias en estudios previos y replicar experimentos utilizando datos en distintos formatos, lo que contribuye a fortalecer la fiabilidad de los resultados obtenidos en investigaciones biomédicas y otras disciplinas científicas [25].

Otro avance significativo ha sido el desarrollo de asistentes de investigación basados en IA, los cuales pueden analizar grandes volúmenes de literatura científica, identificar información relevante y generar resúmenes estructurados que facilitan la exploración de nuevos enfoques metodológicos [25]. Estos sistemas han mejorado la eficiencia de los investigadores al reducir el tiempo requerido para revisar grandes cantidades de documentos y extraer conocimiento relevante de bases de datos científicas [9,29]. Asimismo, los agentes inteligentes han sido implementados en sistemas de recomendación en ciencias biomédicas, donde analizan las características de los datos experimentales y sugieren metodologías estadísticas óptimas, asegurando que los análisis sean apropiados para cada tipo de estudio [26–28].

El avance continuo de los agentes inteligentes, impulsado por la evolución de los LLMs y las redes neuronales avanzadas, permitirá que estos sistemas no solo procesen información de manera eficiente, sino que también desarrollen capacidades de razonamiento autónomo, facilitando la generación de nuevo conocimiento en múltiples disciplinas científicas. Su integración en entornos de investigación continuará optimizando el análisis de datos, mejorando la reproducibilidad de estudios y promoviendo el acceso a

herramientas de análisis avanzadas, lo que representa un cambio paradigmático en la manera en que se realiza la investigación científica en la actualidad [39].

2.6 Estado del Arte o Antecedentes

En las ciencias biomédicas, la selección de la prueba estadística adecuada es un paso crítico y complejo, por lo que la irrupción de modelos de lenguaje extenso (LLM) ha permitido automatizar este proceso, mejorando la eficiencia y precisión en la toma de decisiones. Mondal et al. [115], evaluaron la capacidad de LLM gratuitos (ChatGPT3.5, Google Bard, Microsoft Bing Chat y Perplexity) para recomendar pruebas estadísticas apropiadas, comparando sus sugerencias con las de expertos humanos a partir de 27 casos clínicos basados en escenarios de investigación comunes. Los resultados mostraron concordancias de 85,19 % (ChatGPT3.5 y Perplexity), 77,78 % (Bard) y 96,3 % (Bing Chat), con tasas de aceptación superiores al 95 % en todos los casos; adicionalmente, el coeficiente de correlación intraclase promedio fue de 0,728 (IC 95 %: 0,51–0,86, $p < 0,0001$), y la confiabilidad test-retest varió significativamente entre modelos, lo que evidencia que, aunque la experiencia humana resulta indispensable, los LLM pueden aumentar la velocidad de trabajo en manos de expertos.

A principios de 2025, Busch et al. [116], realizaron un metaanálisis en el que se identificó que los transformadores generativos más utilizados en el ámbito médico se centraron principalmente en GPT-3.5 (53,2 %) y GPT-4 (26,6 %). Estos modelos han sido aplicados en la respuesta a preguntas médicas, generación de información para pacientes y documentación clínica. No obstante, el estudio también destacó limitaciones en el diseño —como la falta de optimización para el dominio médico, la transparencia en los datos y problemas de accesibilidad— y en los resultados, evidenciando irreproducibilidad, inexactitud, falta de comprehensividad, inseguridad y sesgo. Este trabajo proporciona un marco y una taxonomía fundamentales para la implementación y evaluación de LLM en entornos de atención médica.

En otro enfoque, Rakauskas et al. [117] evaluaron el desempeño de ChatGPT-3.5, ChatGPT-4 y Microsoft Copilot en preguntas de autoevaluación de cirugía de la mano,

utilizando 1,000 ítems de exámenes realizados entre 2015 y 2019 por la American Society for Surgery of the Hand. Los resultados indicaron que ChatGPT-4 alcanzó una precisión del 63,4 %, superando a Microsoft Copilot (59,9 %) y a ChatGPT-3.5 (51,6 %), con diferencias estadísticamente significativas, y revelaron que la concordancia de las respuestas variaba según el modelo. Estos hallazgos sugieren que, si bien los LLM pueden potenciar la educación médica, es necesario profundizar en el análisis de su consistencia y confiabilidad en distintos contextos clínicos.

Asimismo, se han empleado LLM para evaluar la viabilidad de generar conjuntos de datos clínicos sintéticos mediante prompting zero-shot con GPT-4º. En este sentido, Barr et al. [112] compararon las propiedades estadísticas de los datos generados con las del VitalDB, un conjunto de datos perioperatorios reales. En la primera fase, GPT-4o generó 6,166 registros a partir de 13 parámetros clínicos, demostrando plausibilidad y consistencia (por ejemplo, en el cálculo del índice de masa corporal). En la segunda fase, al replicar las estadísticas descriptivas del VitalDB, se observó una similitud estadística en el 92,31 % de los parámetros analizados, con un solapamiento del 85,71 % en los parámetros continuos. Estos hallazgos evidencian el potencial de los modelos de lenguaje extenso para producir datos sintéticos realistas que faciliten el acceso a información clínica, preservando la privacidad y reduciendo la dependencia de recursos técnicos y computacionales.

En el ámbito de la radiología y las imágenes diagnósticas se ha evaluado el desempeño de GPT-4 en el diagnóstico de imágenes patológicas. El modelo alcanzó una precisión global de 0,64, demostrando buen desempeño en la identificación de cáncer y en la clasificación de pólipos colorrectales, aunque la consistencia entre evaluaciones fue moderada, lo que sugiere que GPT-4 podría ser una herramienta útil para apoyar los diagnósticos en patología [118]. Por otro lado, un estudio comparó la precisión diagnóstica de ChatGPT-4 con y sin integración de datos visuales en 363 casos clínicos. Los resultados mostraron que la inclusión de imágenes (ChatGPT-4V) no mejoró significativamente la identificación del diagnóstico final, ya que su tasa de inclusión en la lista de diferenciales fue del 85,1 % frente al 87,9 % de la versión solo texto, y su capacidad para identificar el diagnóstico principal fue inferior (44,4 % vs 55,9 %; p = 0,002). Esto indica que, actualmente, ChatGPT-4V depende mayoritariamente de la información textual, lo que evidencia la necesidad de

optimizar la integración multimodal para potenciar su desempeño en diagnósticos clínicos [119].

Finalmente, un estudio comparativo evaluó el desempeño de tres modelos de lenguaje extenso —ChatGPT-4, Gemini y Med-Go— en la toma de decisiones clínicas a partir de 134 casos clínicos en nueve especialidades [120]. Las respuestas, evaluadas por expertos en aspectos diagnósticos, criterios diagnósticos, diagnósticos diferenciales, exámenes y tratamientos, revelaron que Med-Go obtuvo la puntuación mediana más alta, superando significativamente a Gemini ($p < 0,001$). Aunque todos los modelos alcanzaron más del 60 % de la puntuación máxima, las deficiencias, especialmente en el diagnóstico diferencial, subrayan la necesidad de cautela en su uso clínico. Estos hallazgos resaltan el potencial de integrar conocimientos médicos especializados en la formación de LLM para mejorar su precisión y seguridad en la práctica médica.

Capítulo 3: Diseño Metodológico

3.1 Tipo de Estudio

Se planteó un tipo de estudio clasificado como aplicado, dado que implicó la implementación y evaluación controlada de un sistema basado en inteligencia artificial generativa y un motor de inferencia experto. Tuvo un carácter cuantitativo, ya que validó la precisión y el desempeño de los agentes LLMs mediante análisis estadísticos. Además, asumió una orientación de comprobación, al evaluar la precisión y aplicabilidad de los agentes en escenarios reales de investigación. En cuanto a la profundidad, se clasificó como un estudio experimental in silico, dado que buscó determinar cómo los LLMs podían mejorar la reproducibilidad en la estadística biomédica, desarrollándose dentro de un marco de laboratorio, en un entorno de simulación con datos biomédicos reales.

3.2 Procedimiento

3.2.1 Desarrollo de Agentes Especializados

El desarrollo del asistente multiagente se basó en una arquitectura modular compuesta por siete agentes especializados, cada uno con una función definida en el ciclo de generación automatizada de análisis estadístico. Cada agente fue implementado como una instancia de un modelo de lenguaje grande (LLM) configurado mediante *system prompts* específicos, que definieron su rol, los criterios de salida esperados y el formato de respuesta, facilitando así la interoperabilidad entre agentes.

El flujo algorítmico se estructuró en un ciclo iterativo y no lineal que integró planificación, análisis, investigación contextual, generación, verificación, depuración y documentación del código estadístico. Este flujo fue diseñado para incluir de manera activa al usuario humano

como elemento central del proceso (enfoque *Human-in-the-Loop*), permitiendo la clarificación de solicitudes ambiguas, la validación de resultados y la retroalimentación continua.

Los agentes desarrollados fueron:

- **Agente Planificador (Planner Agent)**: encargado de transformar una solicitud general en un conjunto organizado de subtareas, identificando ambigüedades y formulando preguntas de clarificación al usuario.
- **Agente Analizador (Analyzer Agent)**: determinó la necesidad de conocimiento de dominio adicional, sugiriendo documentos técnicos o contextuales a integrar.
- **Agente Investigador (Researcher Agent)**: realizó búsquedas contextuales sobre una base vectorial (FAISS), sintetizando la información relevante para alimentar al codificador.
- **Agente Codificador (Coder Agent)**: generó código estadístico en Python de calidad de producción, alineado con las subtareas, contexto y objetivos del usuario.
- **Agente Verificador (Verifier Agent)**: produjo casos de prueba y retroalimentación técnica sobre el código, evaluando robustez, replicabilidad y claridad.
- **Agente Depurador (Debugger Agent)**: procesó errores reportados por el usuario, modificando el código para corregir fallas o aplicar cambios funcionales.
- **Agente Documentador (Documenter Agent)**: generó documentación técnica estructurada, incluyendo descripciones funcionales, pseudocódigo y ecuaciones relevantes en formato LaTeX.

La interacción entre agentes se organizó mediante un sistema de paso de mensajes estructurados y una memoria de sesión compartida, asegurando coherencia entre etapas y continuidad en la trazabilidad del proceso. Además, se implementó una base de conocimiento persistente a través de FAISS, lo que permitió al sistema "recordar"

documentos de dominio previamente indexados y reutilizar esa información en nuevas solicitudes.

Este desarrollo permitió simular un equipo virtual colaborativo de especialistas en análisis estadístico, coordinado por el usuario, y validó el potencial de los LLMs como herramientas de asistencia técnica de alto nivel en la investigación biomédica.

3.2.2 Integración del Modelo de Lenguaje Generativo

La integración del modelo de lenguaje generativo se realizó mediante la orquestación de múltiples instancias de LLMs especializados, cada una configurada mediante system prompts diseñados cuidadosamente para delimitar su rol funcional, estilo de salida y tipo de interacción. Estos LLMs, basados en la tecnología de OpenAI (GPT-4), fueron gestionados como agentes autónomos que intercambian información de manera secuencial y estructurada, conformando un sistema multiagente colaborativo.

El modelo generativo no se limitó a tareas de generación de texto, sino que participó activamente en todo el ciclo de análisis automatizado. Las principales estrategias de integración fueron:

- **Diseño de prompts contextuales:** Cada agente recibió un prompt de sistema que definía su identidad, objetivo, formato esperado de salida y restricciones específicas. Por ejemplo, el agente Documentador debía incluir pseudocódigo y ecuaciones LaTeX, mientras que el Codificador debía generar scripts en el software solicitado con comentarios y lista explícita de librerías.

- **Gestión de la memoria de sesión:** Se implementó una estructura central de memoria (`session_state_memory`) que conservó la información intercambiada entre agentes, incluyendo la solicitud original del usuario, las respuestas a preguntas

de clarificación, los planes generados, el código en evolución, retroalimentación y documentación. Esto permitió un flujo coherente y trazable en todo el proceso.

- **Interacción secuencial mediante funciones:** La interacción entre el usuario y el sistema, así como entre los agentes, se organizó a través de funciones encapsuladas que llamaban al modelo generativo con entradas preprocesadas y recolectaban salidas estructuradas, listas para ser interpretadas por el siguiente agente.

- **Recuperación de contexto vectorial (RAG):** En combinación con la base de conocimiento vectorial implementada con FAISS, el modelo de lenguaje fue capaz de contextualizar sus respuestas a partir de fragmentos de documentos técnicos previamente cargados por el usuario, optimizando así la pertinencia de los análisis y generando explicaciones más ajustadas al dominio.

- **Control del formato y la calidad de salida:** Se establecieron reglas de validación de la salida generada por cada agente (como estructuras de subtareas, bloques de código, listas de pruebas o secciones de documentación), que garantizaban la estandarización y la capacidad de análisis programático de las respuestas. Esto fue clave para permitir la interoperabilidad entre agentes y su integración posterior con interfaces humanas y visuales, en este caso fue **Streamlit**.

Esta integración de los LLMs como núcleo de razonamiento y generación dentro del sistema multiagente permitió una automatización robusta, interpretativa y flexible, que no solo replicó funciones técnicas avanzadas, sino que también facilitó la comunicación natural con el usuario, mejorando la accesibilidad de los métodos estadísticos en contextos biomédicos.

3.2.3 Métodos de Evaluación

La evaluación del asistente multiagente basado en LLMs se estructuró según los tres objetivos específicos planteados en el estudio, aplicando una combinación de métodos funcionales, heurísticos y de percepción del usuario.

Para alcanzar el objetivo de desarrollar un sistema multiagente compuesto por agentes especializados basados en LLMs que colaboraran en la planificación, análisis, investigación, generación, verificación, depuración y documentación de código estadístico, se aplicó una estrategia de validación funcional centrada en el comportamiento operativo de cada uno de los agentes del sistema y su capacidad para interactuar coordinadamente dentro del ciclo de análisis estadístico automatizado. Los métodos implementados fueron los siguientes:

- **Diseño de casos de uso representativos**: Se construyó un conjunto de escenarios de prueba derivados de tareas estadísticas comunes en la investigación biomédica (pruebas de hipótesis, regresión lineal, modelos de supervivencia, análisis de varianza). Cada caso fue formulado como una solicitud general del usuario, que debía ser descompuesta y gestionada por el sistema en todas sus etapas: desde la planificación inicial hasta la documentación final del análisis.

- **Evaluación estructural del flujo inter-agentes**: Se analizaron los flujos de información entre los distintos agentes mediante seguimiento de mensajes estructurados y trazabilidad de las decisiones generadas. Se verificó que los agentes respetaran su rol funcional, utilizaran adecuadamente la información compartida en la memoria de sesión y produjeran salidas compatibles con los agentes subsiguientes.

- **Verificación del cumplimiento del rol por agente**: Para cada agente, se definieron criterios de validación específicos de acuerdo con el propósito asignado por su prompt de sistema. Por ejemplo, el agente Planificador debía generar subtareas claras y detectar ambigüedades, mientras que el Verificador debía producir retroalimentación sobre sintaxis, ejecución y lógica del código. La evaluación

consideró si la salida de cada agente cumplía con estos criterios funcionales esperados.

- **Análisis de completitud y coherencia de las salidas**: Se revisaron todas las respuestas generadas por los agentes en cada caso de uso, evaluando:
    - La completitud de la información entregada, es decir, si contenía todos los elementos requeridos para su función (código, instrucciones, justificación).
    - La coherencia interna, tanto en el contenido como en el formato, considerando la alineación entre subtareas, código generado, documentación y retroalimentación técnica.
    - La adherencia al formato especificado, comparando las salidas obtenidas con la estructura esperada definida en los prompts, incluyendo estilo de codificación, orden lógico, uso de comentarios y etiquetado de variables.
- **Estrategia iterativa de prueba y mejora del sistema:** El sistema fue sometido a un proceso de evaluación cíclica mediante simulaciones sucesivas de uso, orientadas a detectar fallas en la secuencia operativa entre agentes, inconsistencias en la lógica de procesamiento o deficiencias en la transferencia de información. A partir de estas observaciones, se implementaron ajustes incrementales en los mecanismos de control, flujos de datos y diseño de prompts, con el objetivo de optimizar la coherencia funcional del sistema y reforzar su capacidad de autorregulación.

Estos procedimientos permitieron asegurar que el sistema desarrollado no solo ejecutara los pasos esperados, sino que lo hiciera con consistencia, precisión funcional y conforme a los principios de modularidad, trazabilidad y supervisión humana definidos en el marco metodológico.

Para alcanzar el objetivo de validar la funcionalidad del Asistente Multiagente en términos de precisión técnica, claridad, capacidad de replicación de análisis estadísticos y producción de documentación científica estructurada, se diseñó una estrategia de

evaluación heurística centrada en la valoración funcional del sistema desde una perspectiva técnica. Los métodos implementados fueron los siguientes:

- **Diseño de una escala heurística funcional ad hoc de 15 ítems:** Se construyó un instrumento específico para esta investigación con el objetivo de evaluar de forma estructurada el desempeño funcional del asistente multiagente. La escala se fundamentó en criterios técnicos derivados de buenas prácticas en ciencia de datos, calidad de código y documentación reproducible. Se organizaron 15 ítems distribuidos en dos dimensiones principales:
    - *Funcionalidad externa* (8 ítems), que valoró la calidad del producto generado, incluyendo la correspondencia del análisis con la solicitud, claridad del código, justificación del modelo estadístico, replicabilidad, calidad del preprocesamiento, completitud de la documentación, coherencia lógica y estructura del reporte generado.
    - *Funcionalidad interna* (7 ítems), que evaluó el comportamiento técnico del sistema, considerando aspectos como ejecución sin errores, visualización de resultados, consistencia de formato, navegabilidad entre etapas, posibilidad de refinamiento, soporte a nuevas consultas y eficiencia en la entrega de resultados.
- **Escala de medición:** Cada ítem fue calificado mediante una escala tipo Likert de 5 puntos, cuyos niveles de desempeño fueron: 1 = Deficiente, 2 = Bajo, 3 = Aceptable, 4 = Bueno y 5 = Excelente. Esta escala permitió cuantificar la valoración técnica de forma uniforme y facilitar el análisis posterior.
- **Aplicación por jueces expertos:** La escala fue aplicada por dos evaluadores con experiencia en estadística aplicada y desarrollo de herramientas tecnológicas en contextos biomédicos. Los evaluadores analizaron de forma independiente los análisis estadísticos generados por el sistema a partir de 30 artículos científicos seleccionados, cubriendo una amplia variedad de métodos estadísticos comúnmente utilizados en investigación biomédica (e.g., regresión, análisis de varianza, pruebas de hipótesis, modelos multivariados).

- **Estimación de la confiabilidad interevaluador:** Para verificar la consistencia entre los evaluadores, se empleó el *coeficiente AC1 de Gwet*, adecuado para escalas ordinales y robusto frente a sesgos por prevalencia o distribución asimétrica de respuestas.
- **Análisis descriptivo de las puntuaciones funcionales:** Se calcularon medidas de tendencia central y dispersión (media y desviación estándar) tanto para los ítems individuales como para cada dimensión, con el fin de identificar patrones de comportamiento y áreas críticas o destacadas en el desempeño del sistema.

Esta estrategia permitió una valoración técnica precisa y estructurada del sistema desarrollado, integrando criterios de calidad, claridad, replicabilidad y estabilidad funcional, y generando evidencia válida sobre la capacidad del asistente multiagente para ejecutar análisis estadísticos con estándares técnicos exigibles en el ámbito de la investigación biomédica.

Para alcanzar el objetivo de evaluar la aplicabilidad práctica del sistema en escenarios reales de investigación biomédica, considerando usabilidad, eficacia y satisfacción desde la experiencia del usuario, se diseñó una estrategia metodológica mixta que combinó instrumentos de evaluación estandarizados, análisis estadísticos descriptivos e inferenciales y análisis semántico-emocional del lenguaje. Los métodos implementados fueron los siguientes:

- **Aplicación de la escala System Usability Scale (SUS):** Se utilizó la escala SUS, una herramienta estandarizada de 10 ítems ampliamente validada para medir la usabilidad percibida de sistemas tecnológicos. Cada ítem fue valorado en una escala tipo Likert de 5 puntos (de “Totalmente en desacuerdo” a “Totalmente de acuerdo”), con alternancia entre afirmaciones positivas y negativas. Los puntajes fueron transformados a una escala de 0 a 100, permitiendo su interpretación según rangos de usabilidad establecidos (Pobre (<50), Aceptable (50–68), Bueno (68–80), Excelente (>80)).

- **Diseño de un formulario ampliado para caracterización de experiencia del usuario:** Además de la escala SUS, se recogió información sociodemográfica, nivel de formación académica, experiencia laboral y dominio en el uso de software estadístico. También se evaluaron cinco dimensiones emergentes: eficacia, eficiencia, satisfacción, usabilidad técnica y aprendizaje funcional, cada una valorada mediante ítems específicos en escala Likert de 5 puntos.

- **Muestreo de participantes:** El instrumento fue aplicado a una muestra de 34 usuarios, conformada por profesionales e investigadores del área biomédica, con distintos niveles de experiencia y formación, quienes interactuaron con el asistente multiagente simulando casos de análisis reales.

- **Análisis descriptivo y comparativo:** Se calcularon medidas de tendencia central (mediana, media) y dispersión (rango intercuartílico, desviación estándar) para los puntajes globales y por dimensión. Adicionalmente, se aplicaron pruebas estadísticas inferenciales (como chi cuadrado o test de Fisher) para analizar la relación entre variables como experiencia previa y clasificación de la usabilidad percibida.

- **Análisis semántico-emocional del lenguaje libre:** Las respuestas abiertas de los usuarios fueron procesadas mediante análisis textual en R, utilizando los lexicones NRC y Oplexicon en español para identificar emociones básicas (como confianza, alegría, frustración, sorpresa) y polaridad del lenguaje. Se construyeron redes semánticas para cada dimensión evaluada mediante `tidygraph` y `ggraph`, lo que permitió identificar núcleos de percepción emocional asociados a la experiencia con el sistema.

Este enfoque permitió no solo valorar cuantitativamente la usabilidad del sistema, sino también capturar aspectos cualitativos y afectivos de la experiencia del usuario, proporcionando una evaluación integral de la aplicabilidad práctica del asistente multiagente en contextos reales de investigación biomédica.

3.2.4 Consideraciones Éticas

La implementación de agentes basados en Modelos de Lenguaje Grande (LLMs) en bioestadística implica riesgos mínimos, según la Resolución 8430 de 1993, al manejar datos sensibles de investigaciones biomédicas. Para garantizar la protección de la información, se cumplirán normativas como la Ley 1581 de 2012 y su Decreto 1377 de 2013, estableciendo mecanismos de anonimización, cifrado y acceso restringido. Además, se implementarán controles que eviten la identificación de individuos y aseguren la privacidad de los datos procesados.

Finalmente, es fundamental aclarar la responsabilidad legal del uso de estos sistemas, los cuales deben ser considerados herramientas de apoyo y no sustitutos del juicio profesional. Conforme a la Ley 23 de 1981 y la Ley 1438 de 2011, los resultados generados por los agentes LLMs deberán ser validados por expertos en bioestadística o investigadores. Para garantizar la trazabilidad de las decisiones, se registrarán todas las interacciones del sistema, permitiendo auditorías que aseguren la fiabilidad y rigor metodológico del análisis estadístico automatizado.

Capítulo 4: Resultados

4.1 Desarrollo del Asistente Interactivo Multiagente para Análisis Estadístico Automatizado

El sistema opera como un marco colaborativo donde distintos agentes inteligentes, cada uno con un rol especializado, trabajan en conjunto con un usuario humano para descomponer problemas complejos, investigar soluciones, generar código, verificar su corrección, optimizarlo y documentarlo. La interactividad es clave, permitiendo que el usuario no solo inicie el proceso, sino que también guíe, valide y refine las salidas en cada fase, asegurando que el producto final se alinee precisamente con sus requisitos y expectativas. Este enfoque contrasta marcadamente con los modelos de generación de código puramente autónomos, que a menudo carecen de la capacidad de auto-corrección o de la comprensión profunda de las sutilezas del dominio, al integrar la intuición, el juicio y el conocimiento de dominio humano en un bucle de retroalimentación continuo y dinámico. La motivación subyacente es superar las limitaciones inherentes a los LLM monolíticos en tareas complejas de ingeniería de software, donde la precisión, la robustez y la adherencia a especificaciones no triviales son primordiales.

4.1.1 Arquitectura General del Sistema Multi-Agente

El algoritmo se concibe como un ciclo iterativo y no lineal de planificación, análisis, investigación, codificación, verificación, depuración, optimización y documentación. En este ciclo, el usuario humano no es un mero observador, sino el orquestador y validador principal, ejerciendo control y dirección en cada etapa. La interacción es predominantemente secuencial, reflejando un flujo de trabajo de desarrollo estructurado, pero la arquitectura permite bucles de retroalimentación y saltos hacia atrás en cualquier

etapa. Esta flexibilidad es fundamental para abordar la complejidad inherente a las tareas de desarrollo de software, donde los requisitos pueden evolucionar, los errores pueden surgir en cualquier momento y la retroalimentación es esencial para la convergencia hacia una solución óptima. Este diseño modular y colaborativo permite que cada agente se especialice en una fase particular del ciclo de vida del desarrollo, aprovechando al máximo las fortalezas de los Modelos de Lenguaje Grande (LLM) en tareas específicas como el razonamiento lógico, la síntesis de información o la generación de código, mientras que la coordinación explícita entre ellos y la supervisión humana mitigan sus limitaciones conocidas, como la propensión a "alucinar" o a generar soluciones subóptimas sin guía externa.

El sistema se compone de una serie de Agentes, cada uno implementado como una instancia de un LLM con un system_prompt específico usando el modelo de openai. Este system_prompt es una directriz fundamental que moldea la "personalidad" y el comportamiento del LLM, definiendo su rol, las expectativas sobre su salida y las restricciones de formato. Por ejemplo, un system_prompt para un agente de codificación incluirá instrucciones sobre la calidad del código, el estilo de los comentarios y el formato de las dependencias. Esta ingeniería de prompts es crucial para asegurar que cada agente se adhiera a su función designada y produzca resultados en un formato predefinido, lo que facilita enormemente la interoperabilidad y el traspaso de información estructurada entre ellos. La comunicación entre agentes y con el usuario se realiza exclusivamente a través de mensajes de texto estructurados, lo que garantiza la claridad, la capacidad de parseo programático de la información intercambiada y la trazabilidad de las interacciones, permitiendo que el sistema funcione de manera cohesiva y predecible. La modularidad de los agentes también facilita la experimentación con diferentes modelos de LLM para roles específicos o la incorporación de nuevos agentes para tareas adicionales.

4.1.2 Componentes Clave y Flujo Algorítmico

El sistema consta de siete agentes principales, cada uno con una función distinta y bien definida, operando en una secuencia lógica que puede implicar iteraciones y saltos hacia atrás en función de la retroalimentación del usuario o los resultados de las etapas

subsiguientes. Esta orquestación de agentes simula un equipo de desarrollo de software, donde cada "miembro" aporta su experiencia especializada:

1. Agente Planificador (Planner Agent)
2. Agente Analizador (Analyzer Agent)
3. Agente Investigador (Researcher Agent)
4. Agente Codificador (Coder Agent)
5. Agente Verificador (Verifier Agent)
6. Agente Depurador (Debugger Agent)
7. Agente Documentador (Documenter Agent)

A continuación, se describe en detalle el rol, las entradas, el proceso lógico, las salidas y las interacciones de cada agente dentro del flujo algorítmico, destacando la profundidad de su operación, los principios algorítmicos subyacentes y su contribución integral al proceso general de generación de código.

Cuadro 1.- Punto de entrada del algoritmo principal

```
Entrada:
- Solicitud del usuario (texto libre)

Proceso:
- Inicializa la memoria de sesión.
- Define el estado inicial del código.

Salida:
- Estructura `session_state_memory` con campos clave.

Código:
FUNCTION MultiAgentCodeGenerationProcess(user_initial_request):
   Inicializar session_state_memory
   session_state_memory.current_request = user_initial_request
   session_state_memory.clarifications_provided
   session_state_memory.code_status = "PENDING_GENERATION"
```

4.1.2.1 Agente Planificador (Planner Agent)

Rol: Es el agente inicial y central de la fase de concepción y definición del problema. Su función principal es transformar una solicitud de usuario de alto nivel, a menudo vaga o ambigua, en un plan detallado y ejecutable de subtareas. Crucialmente, también es

responsable de identificar cualquier ambigüedad o falta de detalle en la solicitud original que requiera aclaración por parte del usuario. Este rol es vital para asegurar que el problema se entienda correctamente y de manera exhaustiva desde el principio, minimizando la propagación de errores conceptuales y evitando costosos re-trabajos en etapas posteriores del desarrollo. La calidad del plan inicial impacta directamente la eficiencia y la corrección de todo el proceso subsiguiente.

Entradas: La solicitud inicial del usuario, puede variar desde una descripción concisa de una función hasta un problema complejo que implica múltiples componentes y requisitos. (En iteraciones posteriores) Las respuestas del usuario a las preguntas de clarificación previas. Esta entrada es fundamental para el refinamiento iterativo del plan. El Planificador también recibe el plan actual y las preguntas formuladas anteriormente, lo que le permite mantener un historial conversacional y refinar el plan de forma incremental, convergiendo hacia una comprensión mutua y completa del problema.

Proceso Lógico:

1. **Análisis Profundo de la Solicitud:** El agente realiza un análisis semántico y contextual de la solicitud del usuario para desentrañar el objetivo subyacente, los requisitos funcionales explícitos e implícitos, y los requisitos no funcionales (e.g., rendimiento, seguridad, escalabilidad) que puedan inferirse.
2. **Descomposición Jerárquica y Cadena de Pensamiento (CoT):** Utiliza un proceso de "cadena de pensamiento" (Chain-of-Thought, CoT). Este enfoque permite al LLM articular su razonamiento paso a paso, descomponiendo la solicitud en subtareas más pequeñas, manejables, lógicas y secuenciales. Por ejemplo, una solicitud como "crear una aplicación web para gestionar inventarios con una base de datos de productos y usuarios" podría descomponerse en subtareas como: "1. Diseñar el esquema de la base de datos (productos, usuarios, inventario). 2. Implementar la API REST para operaciones CRUD sobre los datos. 3. Desarrollar la interfaz de usuario para la visualización y edición. 4. Añadir un módulo de autenticación y autorización de usuarios. 5. Configurar el entorno de despliegue." Este proceso implica un razonamiento explícito sobre cómo cada subtarea contribuye al objetivo general, mejorando la coherencia, la completitud y la granularidad del plan.

3. **Evaluación Crítica de Ambigüedades:** Evalúa críticamente cada subtarea y la solicitud general en busca de ambigüedades, detalles faltantes, suposiciones implícitas o interpretaciones múltiples. Por ejemplo, si el usuario pide "procesar un archivo de datos", el Planificador podría identificar la ambigüedad de "¿qué tipo de archivo? (CSV, JSON, Excel)", "¿qué formato de datos se espera dentro del archivo?", "¿qué procesamiento específico se debe aplicar (filtrado, agregación, transformación)?", "¿cuál es el volumen de datos esperado?".
4. **Formulación de Preguntas de Clarificación:** Si se identifican ambigüedades, formula preguntas de clarificación específicas, concisas y no redundantes para el usuario. Estas preguntas están diseñadas para obtener la información precisa y necesaria para eliminar la ambigüedad y permitir una ejecución clara y unívoca de las subtareas. La formulación de preguntas es un arte en sí mismo, buscando ser lo suficientemente directas para obtener la información necesaria sin abrumar al usuario.

Salidas:

- Una sección de "Pensamientos" (Thoughts:): Razonamiento interno del agente, mostrando su proceso de decisión, los pasos de descomposición y la justificación de las preguntas formuladas.
- Una lista estructurada de "Subtareas" (Subtasks), que es el plan desglosado, presentado de forma clara, numerada y secuencial, lista los pasos concretos que deben seguirse.
- Adicional una sección de «*Preguntas de Clarificación*» (*Clarifying Questions*), con una lista de preguntas específicas para el usuario. Si el Planificador determina que la solicitud y el plan son perfectamente claros y no requieren más detalles, indica explícitamente «- No clarifying questions needed», señalando que el plan es suficientemente robusto para proceder a las siguientes fases.

Interacción/Handoffs:

- **Hacia el Usuario:** Presenta el plan y las preguntas de clarificación al usuario a través de la interfaz. La interfaz debe facilitar la comprensión y la respuesta a estas preguntas.

- **Desde el Usuario:** Recibe las respuestas del usuario a las preguntas de clarificación. Estas respuestas son cruciales para cerrar el bucle de retroalimentación.
- **Bucle de Refinamiento:** Si hay preguntas de clarificación, el proceso vuelve al Planificador. Este bucle iterativo de "pregunta-respuesta-refinamiento" continúa hasta que el Planificador determina que el plan es suficientemente claro (es decir, no genera más preguntas de clarificación), lo que garantiza que el Planificador y el usuario converjan en un entendimiento mutuo y detallado del problema antes de invertir recursos en la codificación.

Cuadro 2.- Algoritmo del Agente Planificador: Análisis y Generación de Subtareas

```
Entrada:
- Solicitud del usuario
- Respuestas previas del usuario (si existen)

Proceso:
- Analiza paso a paso la solicitud.
- Divide en subtareas explícitas.
- Detecta ambigüedades o vacíos de información.
- Formula preguntas de clarificación si es necesario.

Salida:
- Lista estructurada de subtareas.
- Lista de preguntas de clarificación.

Codigo:
LOOP:
    planner_input = CONCATENAR("Original Request: ",
                                    session_state_memory.current_request,
                               "Previous Answers: ",
                                    session_state_memory.clarifications_provided)
    planner_response = PLANNER_AGENT.LLAMAR(planner_input)
    session_state_memory.plan = EXTRAER_PLAN(planner_response)
    session_state_memory.clarifying_questions =
         EXTRAER_PREGUNTAS_CLARIFICACION(planner_response)

    IF session_state_memory.clarifying_questions NO VACÍO:
        MOSTRAR preguntas AL usuario
        user_input_answers = OBTENER_ENTRADA_USUARIO("Responde preguntas")
        CONCATENAR respuestas a clarifications_provided
    ELSE:
        BREAK LOOP
```

4.1.2.2 Agente Analizador (Analyzer Agent)

Rol: El Agente Analizador actúa como un "gatekeeper" de información y un evaluador de la necesidad de conocimiento externo. Su función es determinar si se requiere documentación de dominio adicional (conocimiento externo, como manuales de API, especificaciones de protocolos o documentos internos de la empresa) para abordar la solicitud del usuario de manera efectiva. Si es así, sugiere qué documentos específicos podrían ser necesarios para informar a los agentes subsiguientes. Este agente previene que los agentes posteriores operen con información insuficiente o incorrecta, lo que podría llevar a soluciones incompletas o erróneas.

Entradas: La solicitud inicial del usuario, que proporciona el contexto general del problema a resolver.

Proceso Lógico:

1. **Evaluación de Necesidad de Documentos:** El agente razona si la solicitud puede ser completada con el conocimiento general inherente al LLM o si se beneficiaría significativamente de información específica de dominio. Por ejemplo, si la solicitud implica la integración con una API de terceros poco común, la manipulación de formatos de datos propietarios, la implementación de algoritmos específicos de una disciplina científica (e.g., bioinformática, finanzas cuantitativas) o el cumplimiento de normativas específicas de la industria, el Analizador identificará la necesidad de documentación técnica relevante.
2. **Identificación de Documentos:** Si se determina que se necesitan documentos, el Analizador lista los tipos o nombres de documentos relevantes de la manera más específica posible. Ejemplos incluyen "manual de la API de Stripe para pagos", "especificación del protocolo HL7 para datos de salud", "documentación de la base de datos NoSQL Cassandra para modelado de datos", "normativa GDPR para el manejo de datos personales", o "artículo de investigación sobre el algoritmo de optimización X".

Salidas:

- Una sección de "Pensamientos" (Thoughts:): Justificación del agente sobre por qué se necesitan (o no se necesitan) documentos, y cómo la información adicional podría impactar la solución.
- Un indicador NEED_DOCS: Yes/No: Una bandera explícita que comunica la conclusión principal del análisis. Si Yes, una lista de documentos requeridos, cada uno en una nueva línea con un guión, facilitando al usuario la identificación y carga.

Interacción/Handoffs:

- **Hacia el Usuario:** Informa al usuario sobre la necesidad de documentos y los documentos sugeridos. El usuario tiene la opción de cargar estos documentos en la base de datos vectorial del sistema para que el Investigador los utilice. Esta interacción permite al usuario enriquecer el "cerebro" del sistema con conocimiento relevante y específico para la tarea.
- **Hacia el Investigador:** La salida del Analizador influye directamente en si el Investigador realizará una búsqueda activa en la base de conocimiento cargada por el usuario. Si NEED_DOCS es No, el Investigador se basará principalmente en el conocimiento general del LLM y cualquier contexto de archivo general proporcionado.

Cuadro 3.- Algoritmo del Agente Analizador: Evaluación de Necesidad de Conocimiento Externo

```
Entrada:
- Solicitud del usuario

Proceso:
- Evalúa si se requieren documentos de dominio.
- Sugiere nombres de documentos necesarios.

Salida:
- Bandera NEED_DOCS: Yes/No
- Lista de documentos sugeridos

Código:
analyzer_response = ANALYZER_AGENT.LLAMAR(session_state_memory.current_request)
session_state_memory.needs_domain_docs =
EXTRAER_BANDERA_NECESIDAD_DOCS(analyzer_response)
session_state_memory.suggested_docs_list =
EXTRAER_DOCUMENTOS_SUGERIDOS(analyzer_response)

IF session_state_memory.needs_domain_docs OR NOT ES_ÍNDICE_FAISS_CARGADO():
    MOSTRAR documentos sugeridos AL usuario
    uploaded_files = OBTENER_ARCHIVOS_CARGADOS_USUARIO()
    IF uploaded_files NO VACÍO:
        LLAMAR IndexDocumentsIntoFAISS(uploaded_files)
        session_state_memory.faiss_indexed = TRUE
```

4.1.2.3 Agente Investigador (Researcher Agent)

Rol: El Agente Investigador es el responsable de la recopilación y síntesis de información. Recopila datos relevantes de una base de conocimiento vectorial (si está disponible y se ha cargado) y/o de un documento de contexto general proporcionado por el usuario. Su objetivo es proporcionar un resumen estructurado y conciso que contenga la información más pertinente para apoyar la implementación del plan por parte del Codificador. Este agente es crucial para dotar al sistema de conocimiento más allá de sus datos de entrenamiento iniciales, permitiéndole abordar problemas que requieren información actualizada o específica del dominio.

Entradas: La solicitud original del usuario, que establece el objetivo general. El plan de subtareas generado por el Planificador, que detalla los pasos específicos para los cuales se necesita investigación. (Opcional) Texto de un documento de contexto general cargado por el usuario (e.g., un archivo CSV con datos de ejemplo, un extracto de un informe, un manual de usuario simple). (Opcional) Fragmentos de texto (chunks) recuperados de la

base de conocimiento vectorial (FAISS), que son los resultados de búsquedas de similitud realizadas internamente.

Proceso Lógico:

1. **Formulación de Consulta de Búsqueda:** Genera una consulta de embedding combinando la solicitud original del usuario y el plan detallado. Esta combinación ayuda a asegurar que la búsqueda de información sea lo más contextualmente relevante posible para la tarea actual, capturando la intención del usuario y los requisitos del plan.
2. **Recuperación de Información (RAG):** Si existe una base de conocimiento vectorial (FAISS) y fragmentos de texto (chunks) disponibles, el Investigador realiza una búsqueda de similitud. La similitud se calcula utilizando la distancia L2 (Euclidean distance) entre los vectores de embedding:

$$D(v_1, v_2) = \sum_{i=1}^{n} (v_{1i} - v_{2i})^2$$

Ecuación 3.– Formula de la distancia euclidiana entre dos vectores

donde $v1$ es el vector de la consulta y $v2$ es un vector de un fragmento de documento en la base de conocimiento. FAISS (IndexFlatL2) es utilizado para esta búsqueda eficiente, permitiendo recuperar rápidamente los k fragmentos de texto más semánticamente similares a la consulta, incluso en bases de datos de gran tamaño. Para escenarios con millones o miles de millones de vectores, FAISS ofrece índices más avanzados como IndexIVFFlat (que particiona el espacio vectorial) o IndexHNSW (que construye un grafo de vecinos cercanos), que equilibran la velocidad de búsqueda con la precisión. El número de fragmentos k a recuperar se ajusta dinámicamente para evitar sobrecargar el contexto del LLM.
3. **Síntesis de Contexto Integrado:** Combina el texto de los documentos de contexto general cargados por el usuario y los fragmentos recuperados de la base de conocimiento. Este proceso de agregación es vital para proporcionar un contexto completo y coherente a los agentes subsiguientes, eliminando

redundancias y priorizando la información más relevante.

4. **Resumen Estructurado y Destilación:** Sintetiza toda la información contextual en un resumen estructurado. Este resumen no es una simple concatenación de textos, sino una destilación de la información más pertinente, destacando puntos clave, conceptos importantes, posibles desafíos, soluciones conocidas y datos relevantes para la implementación del plan. El Investigador debe ser capaz de identificar y extraer los "insights" clave del material de investigación.

Salidas:

- Una sección de "Pensamientos" (Thoughts:): El proceso de razonamiento del Investigador, incluyendo cómo formuló la consulta, qué tipo de información buscó y por qué ciertos puntos son relevantes.
- Un "Resumen de Investigación" estructurado (Research Summary:), que puede incluir viñetas, párrafos concisos o incluso tablas si la información lo amerita. Este resumen es la entrada principal para el Codificador.

Interacción/Handoffs:

- **Hacia el Codificador:** Proporciona el resumen de investigación al Agente Codificador. Este traspaso es fundamental, ya que el Codificador utilizará esta información para generar un código que no solo sea funcional, sino que también se base en el conocimiento más relevante y actualizado disponible, evitando la generación de soluciones genéricas o desactualizadas.

Cuadro 4.- Algoritmo del Agente Investigador: Síntesis de Información Contextual

```
Entrada:
- Solicitud del usuario
- Plan de subtareas
- Contexto del usuario y documentos recuperados

Proceso:
- Recupera contexto desde FAISS.
- Sintetiza información relevante y estructurada.

Salida:
- Resumen estructurado de investigación

Código:
retrieved_context =
     RECUPERAR_CONTEXTO_DESDE_FAISS(session_state_memory.current_request,
     session_state_memory.plan)
research_prompt = CONCATENAR("Request: ", session_state_memory.current_request,
                             "Plan: ", session_state_memory.plan,
                             "Context: ", retrieved_context,
                             "Extra Context: ",
                                   OBTENER_TEXTO_ARCHIVO_CONTEXTO_PRINCIPAL())
researcher_response = RESEARCHER_AGENT.LLAMAR(research_prompt)
session_state_memory.research_summary =
     EXTRAER_RESUMEN_INVESTIGACION(researcher_response)
```

4.1.2.4 Agente Codificador (Coder Agent)

Rol: El Agente Codificador es el motor de la generación de código. Su función principal es traducir el plan detallado y el resumen de investigación en código Python funcional y de "calidad de producción". Además de la generación inicial, este agente también es responsable de la optimización del código en una fase posterior, lo que implica refinar el código existente para mejorar su rendimiento, legibilidad o adherencia a las mejores prácticas. Su objetivo es producir código que no sólo funcione, sino que sea robusto, mantenible y escalable.

Entradas: La solicitud original del usuario, para mantener la alineación con el objetivo final. El plan de subtareas detallado, que proporciona la estructura y los pasos lógicos a seguir. El resumen de investigación, que proporciona el contexto técnico, los detalles de dominio, los algoritmos específicos o las consideraciones de API necesarias para una implementación correcta. (Opcional) Texto de un documento de contexto general, si es relevante para la lógica de negocio o el formato de datos (e.g., un esquema de CSV o un

formato de mensaje). (En fase de optimización) El código actual que necesita ser mejorado, junto con la retroalimentación detallada del Verificador (incluyendo casos de prueba fallidos, sugerencias de mejora de rendimiento o de estilo).

Proceso Lógico:

1. **Planificación de Implementación Detallada:** Razona paso a paso cómo implementar cada subtarea del plan en código Python. Esto incluye la selección de estructuras de datos apropiadas, la elección de algoritmos eficientes, la consideración de patrones de diseño de software (e.g., modularidad, separación de preocupaciones), y la identificación de cualquier caso extremo (edge cases) que deba ser manejado (e.g., entradas nulas, divisiones por cero, archivos no encontrados). También analiza las dependencias lógicas entre las diferentes partes del código y cómo se integrarán.
2. **Generación de Código de Producción:** Escribe el código Python completo y comentado. El término "calidad de producción" implica que el código debe ser:
    - **Funcional:** Cumple con los requisitos establecidos.
    - **Legible:** Utiliza nombres de variables y funciones claros, sigue convenciones de estilo (e.g., PEP 8), y está bien estructurado.
    - **Robusto:** Incluye manejo de errores (e.g., bloques try-except), validación de entradas y considera posibles fallos.
    - **Modular:** Se divide en funciones y clases lógicas para facilitar el mantenimiento y la reutilización.
    - **Eficiente:** Considera la complejidad algorítmica y el uso de recursos cuando sea relevante.
3. **Identificación y Listado de Dependencias:** Identifica y lista explícitamente todas las librerías Python de terceros requeridas por el código (e.g., pandas, numpy, requests, scikit-learn==0.24.2). Esta lista es crucial para la reproducibilidad del entorno de ejecución y para que el usuario pueda instalar fácilmente las dependencias necesarias.

Salidas:

- Una sección de "Pensamientos" (Thoughts:): El proceso de razonamiento del Codificador, incluyendo las decisiones de diseño, las consideraciones sobre la

implementación de algoritmos y las justificaciones para la elección de librerías.
- Una sección "Librerías Requeridas" (Required Libraries:), con cada librería especificada en una nueva línea, incluyendo opcionalmente versiones específicas para asegurar la compatibilidad.
- El bloque de código Python completo y comentado, encerrado en delimitadores de bloque de código (```python y ` ```), listo para ser ejecutado o revisado.

Interacción/Handoffs:

- **Hacia el Usuario:** Presenta el código generado al usuario para su revisión, ejecución y validación.
- **Hacia el Verificador:** Su código es la entrada principal para el Agente Verificador, quien generará pruebas y retroalimentación sobre su corrección y calidad.
- **Hacia el Depurador:** Su código es la entrada principal para el Agente Depurador si el usuario encuentra errores de tiempo de ejecución o solicita modificaciones específicas.
- **Bucle de Optimización:** Recibe la retroalimentación del Verificador (a través del usuario) para generar una versión optimizada del código. Este bucle permite un refinamiento continuo del código, mejorando su rendimiento, legibilidad y eficiencia hasta alcanzar un estado deseable.

Cuadro 5.- Algoritmo del Agente Codificador: Generación de Código y Dependencias

```
Entrada:
- Solicitud
- Plan
- Resumen de investigación
- Contexto adicional

Proceso:
- Genera código Python de calidad de producción.
- Lista librerías necesarias.

Salida:
- Código en Python
- Requisitos del código (librerías)

Código:
coder_prompt = CONCATENAR("Request: ", session_state_memory.current_request,
                          "Plan: ", session_state_memory.plan,
                          "Research: ", session_state_memory.research_summary,
                          "Context: ", OBTENER_TEXTO_ARCHIVO_CONTEXTO_PRINCIPAL())
coder_response = CODER_AGENT.LLAMAR(coder_prompt)
session_state_memory.current_code = EXTRAER_CODIGO(coder_response)
session_state_memory.code_requirements = EXTRAER_REQUISITOS(coder_response)
session_state_memory.code_status = "GENERATED"
```

4.1.2.5 Agente Verificador (Verifier Agent)

Rol: El Agente Verificador es el garante de la calidad y la corrección del código. Su función es generar casos de prueba exhaustivos para el código producido por el Codificador, incluyendo casos de borde y escenarios de fallo. Además, proporciona retroalimentación constructiva sobre posibles problemas, ineficiencias o áreas de mejora en el código. Este agente es fundamental para la fase de aseguramiento de calidad automatizada.

Entradas: El código Python generado por el Codificador, que es el artefacto para verificar.

Proceso Lógico:

1. **Análisis de Código y Requisitos:** Examina el código para comprender su lógica interna, sus funciones, sus entradas y salidas esperadas, y los posibles flujos de ejecución. Crucialmente, también infiere los requisitos funcionales y no funcionales implícitos en el código y en el plan original para diseñar pruebas relevantes.
2. **Generación de Casos de Prueba Estratégicos:** Diseña un conjunto de casos

de prueba que cubren la funcionalidad principal (casos de uso típicos), así como los casos de borde (e.g., valores mínimos/máximos para entradas numéricas, cadenas vacías, listas vacías, entradas nulas, divisiones por cero, límites de memoria o tiempo si son relevantes). También genera casos negativos (entradas inválidas que deberían generar errores controlados o comportamientos específicos) para probar la robustez. Para cada caso de prueba, especifica claramente las entradas (input=...) y las salidas esperadas (expected=...).

3. **Identificación de Problemas y Sugerencias de Mejora:** Analiza el código en busca de posibles errores lógicos, ineficiencias algorítmicas (e.g., complejidad de tiempo o espacio subóptima), vulnerabilidades de seguridad (aunque de forma limitada para un LLM), problemas de legibilidad (e.g., código spaghetti, falta de comentarios, nombres poco claros) o falta de adherencia a las mejores prácticas de codificación. La retroalimentación puede incluir sugerencias para refactorización, optimización de rendimiento, mejora del manejo de errores o simplificación de la lógica.

Salidas:

- Una sección de "Pensamientos" (Thoughts:): El proceso de razonamiento del Verificador, explicando cómo abordó la verificación, qué tipos de pruebas consideró y la justificación de su retroalimentación.
- Una lista de "Casos de Prueba" (Test Cases:), con entradas y salidas esperadas para cada escenario, presentados de forma clara y ejecutable conceptualmente.
- Una sección de "Retroalimentación" (Feedback:), con problemas o mejoras sugeridas en formato de viñetas, proporcionando puntos de acción claros para el Codificador o el usuario.

Interacción/Handoffs:

- **Hacia el Usuario:** Presenta los casos de prueba y la retroalimentación al usuario. El usuario puede utilizar estos casos de prueba para ejecutar manualmente el código y verificar su comportamiento, o para solicitar directamente una depuración o una optimización.
- **Hacia el Codificador (para optimización):** Su retroalimentación es una entrada crítica para el Agente Codificador en la fase de optimización. El Codificador utilizará

esta información para refinar el código, corrigiendo problemas identificados y aplicando las mejoras sugeridas.

Cuadro 6.- Algoritmo del Agente Depurador: Diagnóstico y Corrección de Código

```
Entrada:
- Código actual con errores o requerimientos de cambio.
- Descripción del error o petición del usuario.

Proceso:
- Analiza paso a paso el error y su causa.
- Genera nuevo código corregido.
- Actualiza lista de librerías si corresponde.

Salida:
- Código corregido en Python
- Requisitos actualizados

Código:
debugger_prompt = CONCATENAR("Current Code: ", session_state_memory.current_code,
                             "Error/Change Request: ",
session_state_memory.user_debug_request,
                             "Original Request: ",
session_state_memory.current_request)
debugger_response = DEBUGGER_AGENT.LLAMAR(debugger_prompt)
session_state_memory.current_code = EXTRAER_CODIGO(debugger_response)
session_state_memory.code_requirements = EXTRAER_REQUISITOS(debugger_response)
session state memory.code status = "GENERATED"
```

4.1.2.6 Agente Depurador (Debugger Agent)

Rol: El Agente Depurador es el especialista en la resolución de problemas. Su función principal es modificar y corregir el código Python existente basándose en un mensaje de error de tiempo de ejecución (que el usuario obtiene al ejecutar el código) o una solicitud de cambio funcional o de estilo del usuario. Este agente es fundamental para la iteración rápida y la resiliencia del sistema, permitiendo que el proceso de desarrollo continúe incluso frente a errores iniciales.

Entradas:

- El código Python actual que presenta el problema. Es crucial que sea el código completo para que el Depurador tenga todo el contexto.
- Un mensaje de error de tiempo de ejecución (e.g., un traceback de Python) o una descripción detallada de un problema funcional (e.g., "la función X no devuelve el

valor correcto para la entrada Y") o un cambio deseado por el usuario (e.g., "cambiar el formato de salida a JSON", "añadir un parámetro para Z").
- La solicitud original del usuario (para contexto, asegurando que la corrección o modificación se alinee con el objetivo inicial del proyecto).

Proceso Lógico:

1. **Diagnóstico del Problema y Análisis de Causa Raíz:** Analiza el código proporcionado y el mensaje de error/descripción del problema. Esto implica una comprensión profunda de la sintaxis y semántica de Python, la lógica del programa y la interpretación de los tracebacks. El agente rastrea el flujo de ejecución para pinpoint la causa raíz del error, que podría ser un error de sintaxis, un error lógico, un problema de manejo de datos, una dependencia faltante o un uso incorrecto de una librería.
2. **Planificación de la Corrección/Modificación:** Formula un plan paso a paso para corregir el error o implementar el cambio solicitado. Este plan considera la mínima intervención necesaria para resolver el problema sin introducir nuevos errores o efectos secundarios no deseados. Busca soluciones elegantes y eficientes.
3. **Modificación del Código In Situ:** Aplica las modificaciones necesarias directamente al código original. Es crucial que el Depurador devuelva el *código completo* con la corrección integrada, no solo el fragmento modificado. Esto evita problemas de integración y asegura que el usuario siempre reciba un script ejecutable completo.
4. **Actualización de Dependencias:** Revisa y actualiza la lista de librerías requeridas si los cambios introducidos lo ameritan (e.g., si la corrección implica el uso de una nueva librería, o si una versión específica es necesaria).

Salidas:

- Una sección de "Pensamientos" (Thoughts:): El proceso de razonamiento del Depurador, incluyendo su análisis del error, la causa raíz identificada y el plan de acción para la corrección o modificación.
- Una sección "Librerías Requeridas" (Required Libraries:), actualizada con cualquier cambio en las dependencias.

- El bloque de código Python completo y corregido/modificado, encerrado en delimitadores de bloque de código.

Interacción/Handoffs:

- **Hacia el Usuario:** Presenta el código corregido/modificado al usuario para su revisión y nueva ejecución.
- **Bucle de Depuración:** Permite un ciclo iterativo de depuración con el usuario. El usuario ejecuta el código, proporciona nueva retroalimentación (si el problema persiste o surge uno nuevo), y el Depurador itera hasta que el código funcione correctamente, lo que es esencial para la robustez y la usabilidad del sistema.

Cuadro 7.- Algoritmo del Agente Verificador: Generación de Pruebas y Retroalimentación

```
Entrada:
- Código aprobado

Proceso:
- Genera casos de prueba relevantes
- Ofrece retroalimentación sobre calidad y mejoras

Salida:
- Casos de prueba
- Comentarios de verificación

Código:
verifier_response = VERIFIER_AGENT.LLAMAR(session_state_memory.current_code)
session_state_memory.test_cases = EXTRAER_CASOS_PRUEBA(verifier_response)
session_state_memory.verifier_feedback = EXTRAER_RETROALIMENTACION(verifier_response)
```

4.1.2.7 Agente Documentador (Documenter Agent)

Rol: El Agente Documentador es el encargado de la fase final del ciclo de desarrollo, asegurando que el código producido sea comprensible y mantenible. Su función es generar documentación completa y de alta calidad para el código final, incluyendo una visión general, descripciones detalladas de funciones, y, si aplica, pseudocódigo para los traspasos entre agentes y ecuaciones en formato LaTeX para conceptos matemáticos o algorítmicos. Este agente es crucial para la longevidad del software y la colaboración futura.

Entradas:

- La solicitud original del usuario, para proporcionar contexto al propósito del código.
- El plan final (posiblemente refinado), que describe la estructura lógica del proyecto.
- El resumen de investigación, que aporta el conocimiento de dominio y técnico subyacente.
- El código Python final (posiblemente optimizado y depurado), que es el artefacto principal a documentar.
- La lista de librerías requeridas, para incluir en la sección de dependencias.

Proceso Lógico:

1. **Análisis Integral del Código y Contexto:** Comprende el diseño general del código, la interacción entre sus componentes, las funciones clave, los algoritmos utilizados y la lógica de negocio implementada. También integra el contexto de la solicitud original, el plan y la investigación para crear una documentación coherente y completa.
2. **Generación de Documentación Estructurada:** Produce una documentación estructurada en formato Markdown, que es fácilmente legible y convertible a otros formatos. La documentación incluye:
    - **Visión General:** Una descripción de alto nivel del propósito del código, su arquitectura general y su funcionalidad principal. Explica el "qué" y el "por qué" del software.
    - **Descripciones Detalladas de Funciones/Clases:** Para cada función o clase significativa, proporciona una explicación detallada de su propósito, los parámetros que acepta (con sus tipos y descripciones), lo que devuelve, cualquier efecto secundario, y ejemplos de uso si son pertinentes. Esto es análogo a la generación de docstrings de Python.
    - **Pseudocódigo de Traspasos de Agentes/Flujos de Trabajo:** Si el código es parte de un sistema multi-agente o interactúa con otros componentes complejos, describe cómo interactúan los componentes a un nivel algorítmico utilizando pseudocódigo. Esto es útil para entender la orquestación de sistemas más grandes.
    - **Ecuaciones en LaTeX:** Representa cualquier concepto matemático o algorítmico relevante utilizando la sintaxis LaTeX para una presentación

clara, precisa y profesional. Por ejemplo, para describir un algoritmo de búsqueda de similitud en un espacio vectorial, podría incluir la fórmula:

$$Similitud\ del\ coseno(A, B) = \frac{A * B}{\|A\|\|B\|}$$

Ecuación 4.– Similitud del coseno entre dos vectores

donde $A * B$ es el producto punto de los vectores A y B, y $|A|$ y $|B|$ son sus magnitudes (norma euclidiana). O para un proceso iterativo que converge a una solución, como en optimización numérica o aprendizaje automático:

$$x_{n+1} = G(x_n, \nabla L(x_n), parámetros)$$

Ecuación 5.– Fórmula general de actualización iterativa en aprendizaje automático

donde x_n es el estado actual, $\nabla L(x_n)$es el gradiente de una función de pérdida, y G es una función de actualización.

Salidas:

- Una sección de "Pensamientos" (Thoughts:): El proceso de razonamiento del Documentador, incluyendo cómo estructuró la documentación y qué aspectos consideró más importantes para resaltar.
- La documentación completa en formato Markdown, con ecuaciones en LaTeX correctamente delimitadas.

Interacción/Handoffs:

- **Hacia el Usuario:** Proporciona la documentación final y sirve como base para preguntas y respuestas posteriores sobre el código. Esta documentación es un activo valioso para el usuario, facilitando la comprensión, el mantenimiento y la futura evolución del código.

Cuadro 8.- Algoritmo del Agente Documentador: Generación de Documentación Técnica y Funcional

```
Entrada:
- Solicitud, plan, resumen, código final y requisitos

Proceso:
- Redacta documentación estructurada (overview, funciones, pseudocódigo, ecuaciones)

Salida:
- Documentación en Markdown

Código:
documenter_prompt = CONCATENAR("Request: ", session_state_memory.current_request,
                               "Plan: ", session_state_memory.plan,
                               "Research: ", session_state_memory.research_summary,
                               "Code: ", session_state_memory.current_code,
                               "Requirements: ",
session_state_memory.code_requirements)
documenter_response = DOCUMENTER_AGENT.LLAMAR(documenter_prompt)
session state memory.documentation = EXTRAER DOCUMENTACION(documenter response)
```

Cuadro 9.- Algoritmo de Q&A: Pregunta sobre documentación/código

```
Entrada:
- Pregunta del usuario
- Documentación y código

Proceso:
- Genera explicación estructurada basada en contexto disponible

Salida:
- Respuesta explicativa al usuario

Código:
LOOP:
    user_qa_question = OBTENER_ENTRADA_USUARIO("Pregunta sobre documentación/código:")
    IF user_qa_question NO VACÍO:
        qa_answer = GENERAR_RESPUESTA_QA(session_state_memory.documentation,
                                         session_state_memory.current_code,
                                         user_qa_question)
        MOSTRAR qa_answer AL usuario
    ELSE:
        BREAK LOOP
```

4.1.2.8 Gestión del Conocimiento y Bases de Datos Vectoriales

El sistema incorpora un componente robusto de gestión del conocimiento mediante una base de datos vectorial persistente, implementada con FAISS (faiss.IndexFlatL2). Esta capacidad es fundamental para que el sistema pueda aprender de la información

proporcionada por el usuario (conocimiento de dominio específico) y aplicar ese conocimiento en tareas futuras, trascendiendo las limitaciones de los datos de entrenamiento estáticos de los LLM. Este mecanismo es una forma de "memoria" a largo plazo para el sistema.

Indexación de Documentos: Cuando el usuario carga documentos de dominio (PDF, DOCX, CSV, TXT), el sistema los procesa meticulosamente para construir y enriquecer su base de conocimiento:

1. **Extracción de Texto:** Se extrae el texto sin formato de los documentos. Este paso maneja diversos formatos de archivo (PDF, DOCX, CSV, TXT) para obtener el contenido textual puro, utilizando librerías como PyPDF2 y python-docx. Se implementan mecanismos de fallback para la decodificación de caracteres (e.g., UTF-8 a Latin-1) para maximizar la recuperación de texto.
2. **Segmentación (Chunking):** El texto extraído se divide en fragmentos (chunks) de tamaño limitado (e.g., un max_tokens de 2048 tokens con un overlap de 256 tokens) utilizando un tokenizador específico (tiktoken, cl100k_base). El solapamiento es una característica crucial: asegura que el contexto no se pierda abruptamente en los límites de los fragmentos y que la información relevante sea capturada en múltiples chunks. Esto aumenta la probabilidad de que una consulta de búsqueda recupere todos los fragmentos necesarios para comprender un concepto completo. La elección del tamaño del chunk y el solapamiento es un hiperparámetro crítico que afecta tanto la granularidad de la recuperación como la cantidad de contexto que un LLM puede procesar eficientemente.
3. **Generación de Embeddings:** Se generan embeddings vectoriales de alta dimensión para cada fragmento utilizando un modelo de embedding de OpenAI (e.g., text-embedding-3-large). Los embeddings son representaciones numéricas que capturan el significado semántico del texto; fragmentos de texto con significados similares tendrán vectores cercanos en el espacio de embedding. Este proceso se realiza en lotes para optimizar la eficiencia de la API, con manejo de errores y reintentos para elementos individuales si un lote falla, lo que mejora la robustez del proceso de indexación.
4. **Almacenamiento en FAISS:** Estos embeddings se añaden a un índice FAISS.

FAISS (Facebook AI Similarity Search) es una librería de código abierto optimizada para la búsqueda eficiente de similitud y el agrupamiento de vectores densos en grandes colecciones. IndexFlatL2 es el tipo de índice más simple, que realiza una búsqueda de fuerza bruta de los vecinos más cercanos utilizando la distancia euclidiana. Para conjuntos de datos con millones o miles de millones de vectores, FAISS ofrece índices más avanzados como IndexIVFFlat (que particiona el espacio vectorial en clústeres y busca solo en los más relevantes) o IndexHNSW (Hierarchical Navigable Small World, que construye un grafo de vecinos cercanos para una búsqueda ultra-rápida), que equilibran la velocidad de búsqueda con la precisión. Los fragmentos de texto originales correspondientes se almacenan en una lista paralela, manteniendo la correspondencia biunívoca entre el vector y su contenido textual original.

5. **Persistencia del "Cerebro":** El índice FAISS y los fragmentos de texto se guardan en disco (brain.index, brain_chunks.pkl) utilizando faiss.write_index y pickle.dump respectivamente. Esto permite que el "cerebro" del sistema persista entre sesiones, eliminando la necesidad de reindexar los documentos cada vez que se inicia la aplicación. Esta persistencia es vital para la acumulación de conocimiento a largo plazo y para que el sistema "recuerde" información específica del usuario o del dominio a lo largo del tiempo.

Recuperación de Información (Retrieval-Augmented Generation - RAG): Durante la fase de investigación, el Agente Investigador aprovecha esta base de conocimiento. Genera un embedding de la consulta (que combina la solicitud del usuario y el plan actual). Esta consulta se utiliza para buscar los fragmentos más relevantes en el índice FAISS. Los fragmentos recuperados se proporcionan al Investigador como contexto adicional para su resumen. Este patrón RAG es fundamental porque permite a los LLM acceder a información externa y actualizada que no estaba en sus datos de entrenamiento, lo que reduce las "alucinaciones" y mejora la precisión y la relevancia de las respuestas generadas. La capacidad de inyectar conocimiento de dominio específico en el contexto del LLM es un diferenciador clave de este sistema.

4.1.2.9 El Rol Crítico del HITL

Una característica fundamental y deliberada de este algoritmo es la integración profunda y continua del usuario humano en cada etapa del proceso de generación de código. El usuario no es un mero observador pasivo que recibe un producto final, sino un participante activo e indispensable que co-crea la solución. Esta interacción forma una sinergia poderosa entre la capacidad de procesamiento masivo y generación de los LLM y la intuición, el juicio crítico, el conocimiento de dominio específico y la capacidad de validación del ser humano. Esta colaboración es lo que permite al sistema abordar problemas complejos y matizados que un solo LLM, operando de forma autónoma, podría tener dificultades para resolver de manera óptima o sin errores, ya que los modelos actuales, a pesar de su impresionante capacidad, pueden "alucinar" (generar información incorrecta pero plausible), producir soluciones subóptimas o malinterpretar requisitos sin una guía externa explícita.

El usuario humano desempeña los siguientes roles críticos, actuando como un supervisor inteligente y un motor de refinamiento:

- **Inicia y Refina la Solicitud:** El usuario proporciona la solicitud inicial, que es el punto de partida para todo el proceso. Esta solicitud puede ser de alto nivel, y es aquí donde la interacción humana se vuelve crucial. Más importante aún, el usuario responde a las preguntas de clarificación del Planificador. Esta interacción iterativa de "pregunta-respuesta" permite al usuario dirigir activamente el proceso, asegurando que la comprensión del problema por parte del sistema sea precisa, completa y alineada con su visión. Esta fase de clarificación es un mecanismo de control de calidad temprana que previene la desviación del objetivo.
- **Proporciona Contexto y Conocimiento de Dominio:** El usuario tiene la capacidad de subir documentos de dominio específicos (e.g., especificaciones técnicas internas, manuales de productos propietarios, datos de ejemplo con formatos únicos). Estos documentos son esenciales para enriquecer la base de conocimiento del sistema con información propietaria, especializada o muy reciente que no estaría disponible en los datos de entrenamiento generales de los LLM. Esta capacidad permite al sistema operar eficazmente en nichos de conocimiento específicos de una empresa o industria, lo que es inalcanzable para un LLM genérico.
- **Valida y Dirige el Flujo:** El usuario revisa críticamente las salidas de cada agente:

el plan generado, el resumen de investigación, el código producido, los casos de prueba y la documentación. Esta validación humana es crucial para detectar errores lógicos, imprecisiones, ineficiencias o desviaciones de los requisitos que los agentes autónomos podrían pasar por alto. El usuario puede decidir si un plan es adecuado, si la investigación es suficiente, si el código es correcto o si la documentación es clara, dirigiendo el sistema a la siguiente fase o a una iteración de refinamiento.

- **Depura y Modifica Directamente:** El usuario es la fuente principal de retroalimentación de depuración. Proporciona mensajes de error de tiempo de ejecución (stack traces) que solo pueden obtenerse al ejecutar el código en un entorno real. Alternativamente, el usuario puede solicitar cambios funcionales o de estilo específicos, basándose en su experiencia o en requisitos cambiantes. Esta retroalimentación activa el Agente Depurador, permitiendo un ciclo de depuración iterativo y eficiente hasta que el código sea funcional, robusto y cumpla con las expectativas operativas.
- **Aprueba y Finaliza:** El usuario indica explícitamente cuándo el código es funcional, cumple con todos los requisitos o cuándo la documentación es satisfactoria. Esta aprobación humana es el punto de control final que permite el avance a la siguiente etapa del flujo de trabajo (e.g., despliegue) o la finalización del ciclo de desarrollo, garantizando que el producto final sea aceptable y útil para el usuario o para su propósito previsto.

4.2 Evaluación heurística de cumplimiento funcional Asistente Multiagente IA

La evaluación general del sistema mediante el coeficiente de concordancia AC1 de Gwet muestra un desempeño sólido en términos de acuerdo entre evaluadores. El promedio global del AC1 para los 15 criterios funcionales fue de 0.831±0.075, lo que indica un nivel de concordancia muy buena y consistente entre los dos evaluadores. Esta baja dispersión sugiere que los criterios fueron interpretados y aplicados de forma estable, reflejando una adecuada claridad de los reactivos y una homogeneidad en la percepción de la funcionalidad del Asistente Multiagente IA en los artículos analizados (Tabla 1).

Tabla 1.– Evaluación heurística funcional del Asistente Multiagente IA basada en la concordancia entre evaluadores y desempeño

Dimensión	AC1	IC_AC1_95	Concordancia	Media_DE	IC_Media_95
Funcionalidad Externa – Validación científica					
Correspondencia	0.701	[0.388–1.014]	Buena	4.53 ± 0.51	[4.36–4.69]
Replicación	0.906	[0.804–1.008]	Muy Buena	3.33 ± 0.69	[3.1–3.55]
Identificacion_variables	0.759	[0.486–1.032]	Buena	1.85 ± 0.66	[1.64–2.06]
Modelo_estadistico	0.790	[0.478–1.102]	Buena	4.08 ± 0.94	[3.77–4.38]
Preprocesamiento	0.732	[0.454–1.01]	Buena	4.85 ± 0.36	[4.73–4.97]
Claridad_codigo	0.808	[0.58–1.036]	Muy Buena	4.88 ± 0.33	[4.77–4.98]
Documentacion	0.936	[0.808–1.064]	Muy Buena	4.88 ± 0.33	[4.77–4.98]
Tiempo_validacion	0.853	[0.644–1.062]	Muy Buena	4.8 ± 0.41	[4.67–4.93]
Ambigüedades	0.817	[0.568–1.066]	Muy Buena	3.65 ± 0.48	[3.5–3.8]
Utilidad_asistente	0.853	[0.644–1.062]	Muy Buena	4.8 ± 0.41	[4.67–4.93]
Funcionalidad Interna - Usabilidad del asistente					
Ejecucion_fuera_Streamlit	0.878	[0.702–1.054]	Muy Buena	4.9 ± 0.3	[4.8–5]
Sin_trazas_consola	0.707	[0.397–1.017]	Buena	3.42 ± 0.5	[3.26–3.59]
Refinamiento_post_ejecucion	0.866	[0.674–1.058]	Muy Buena	4.85 ± 0.36	[4.73–4.97]
Visualizacion_interfaz	0.942	[0.826–1.058]	Muy Buena	4.92 ± 0.27	[4.84–5.01]
Preguntas_codigo_doc	0.936	[0.808–1.064]	Muy Buena	4.88 ± 0.33	[4.77–4.98]

Correspondencia: Correspondencia del código generado con los métodos del artículo; **Replicación:** Replicación de resultados (tablas, figuras, coeficientes); **Identificacion_variables:** Identificación automática de variables clave; **Modelo_estadistico:** Elección adecuada del modelo estadístico; **Preprocesamiento:** Inclusión de pasos de preprocesamiento; **Claridad_codigo:** Claridad del código generado; **Documentacion:** Documentación técnica generada; **Tiempo_validacion:** Tiempo de evaluación/validación reducido; **Ambigüedades:** Manejo de ambigüedades; **Utilidad_asistente:** Utilidad percibida del asistente para automatizar la validación científica; **Ejecucion_fuera_Streamlit:** El código generado puede ejecutarse y validarse fácilmente fuera del entorno Streamlit; **Sin_trazas_consola:** El sistema no muestra trazas innecesarias por consola; **Refinamiento_post_ejecucion:** El sistema permite continuar con refinamiento del código o documentación tras la ejecución; **Visualizacion_interfaz:** Visualización del código generado en la interfaz; **Preguntas_codigo_doc:** Preguntas sobre el código y documentación

En el análisis específico de la Funcionalidad Externa – Validación científica, que agrupa los primeros 10 criterios de evaluación, se observa un promedio de AC1: 0,8155±0,0742, lo que representa un nivel de concordancia buena a muy buena entre los evaluadores. Los aspectos donde la concordancia fue mayor incluyen la documentación técnica generada (AC1: 0.936 [IC95% 0.808–1.064]) y Replicación de resultados (tablas, figuras, coeficientes) (0.906 [IC95% 0.804–1.008]). lo que evidencia un alto grado de acuerdo respecto a la calidad explicative de las evaluaciones realizadas (Tabla 1).

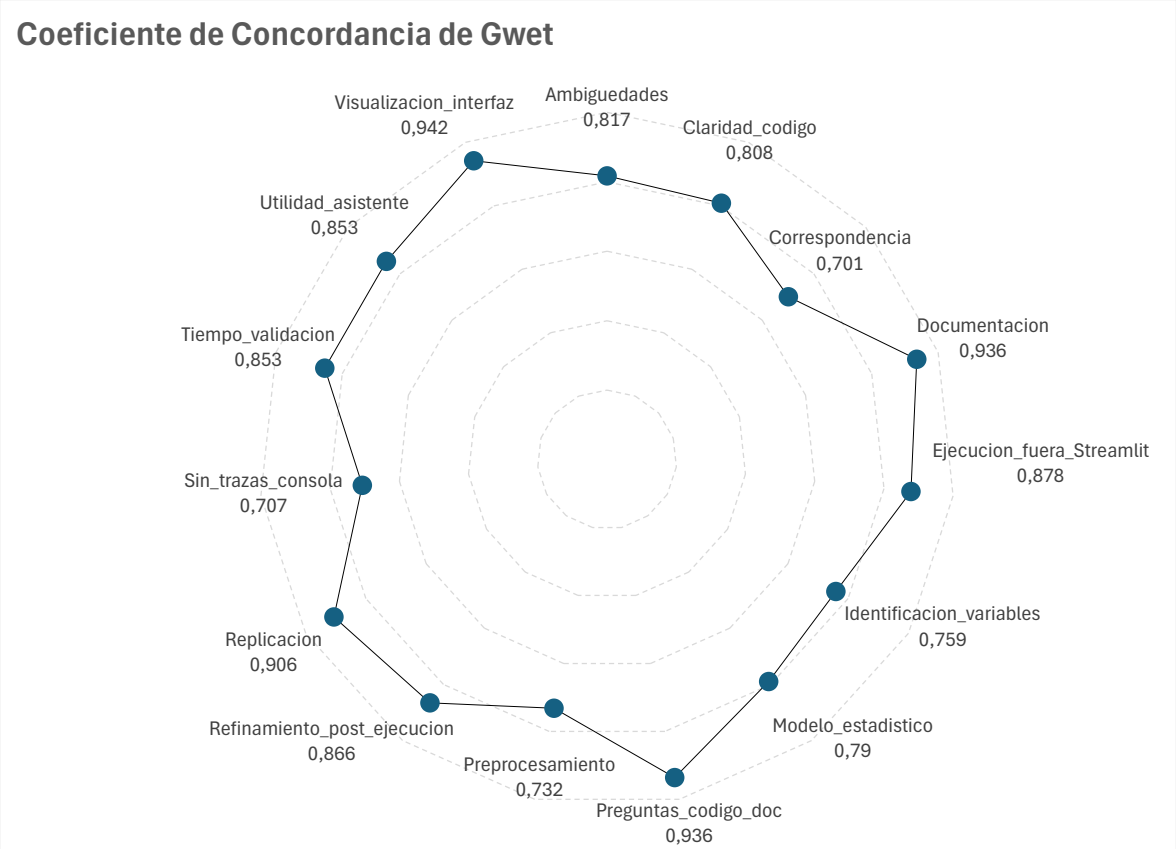


Figura 27.– Coeficiente de concordancia AC1 de Gwet por criterio evaluado en la escala heurística

Correspondencia: Correspondencia del código generado con los métodos del artículo; **Replicacion:** Replicación de resultados (tablas, figuras, coeficientes); **Identificacion_variables:** Identificación automática de variables clave; **Modelo_estadistico:** Elección adecuada del modelo estadístico; **Preprocesamiento:** Inclusión de pasos de preprocesamiento; **Claridad_codigo:** Claridad del código generado; **Documentacion:** Documentación técnica generada; **Tiempo_validacion:** Tiempo de evaluación/validación reducido; **Ambiguedades:** Manejo de ambigüedades; **Utilidad_asistente:** Utilidad percibida del asistente para automatizar la validación científica; **Ejecucion_fuera_Streamlit:** El código generado puede ejecutarse y validarse fácilmente fuera del entorno Streamlit; **Sin_trazas_consola:** El sistema no muestra trazas innecesarias por consola; **Refinamiento_post_ejecucion:** El sistema permite continuar con refinamiento del código o documentación tras la ejecución; **Visualizacion_interfaz:** Visualización del código generado en la interfaz; **Preguntas_codigo_doc:** Preguntas sobre el código y documentación

En contraste, aunque aún dentro de un rango aceptable, los criterios con menor concordancia fueron Correspondencia del código generado con los métodos del artículo (AC1 = 0.701 [IC95% 0.388–1.014]) y preprocesamiento (AC1 = 0.732 [IC95% 0.388–1.014]), lo que podría reflejar variabilidad en cómo los evaluadores interpretaron si el código seguía fielmente la metodología o incluía adecuadamente los pasos de transformación de datos. No obstante, ninguno de los ítems se ubicó en niveles bajos de concordancia, lo que refuerza la fiabilidad del sistema de evaluación (Tabla 1 y Figura 27).

En cuanto a la Funcionalidad Interna – Usabilidad del asistente, compuesta por los últimos cinco criterios de evaluación, se observó una concordancia particularmente sólida, con un promedio de AC1: 0,8658±0,0950. Estos resultados reflejan una muy buena concordancia entre evaluadores, lo que indica consistencia en la evaluación realizada. Los ítems de mayor concordancia fueron Visualización del código en la interfaz (AC1: 0.942 [IC95% 0.826–1.058]) y Preguntas sobre el código y documentación (AC1: 0.936 [IC95% 0.808–1.064]), evidenciando que los evaluadores tenían la misma percepción en cuanto a consultas sobre documentación técnica. Estas características favorecen el aprendizaje autónomo, la revisión crítica y la transparencia del proceso de generación automática. El único ítem con concordancia relativamente menor fue ausencia de trazas innecesarias en consola (AC1 = 0.707 [IC95% 0.397–1.017]), aunque sigue estando dentro del umbral de "buena concordancia". Esto podría indicar diferencias en la percepción del nivel de limpieza o ruido en la salida del Sistema por parte de los evaluadores (Tabla 1 y Figura 27)

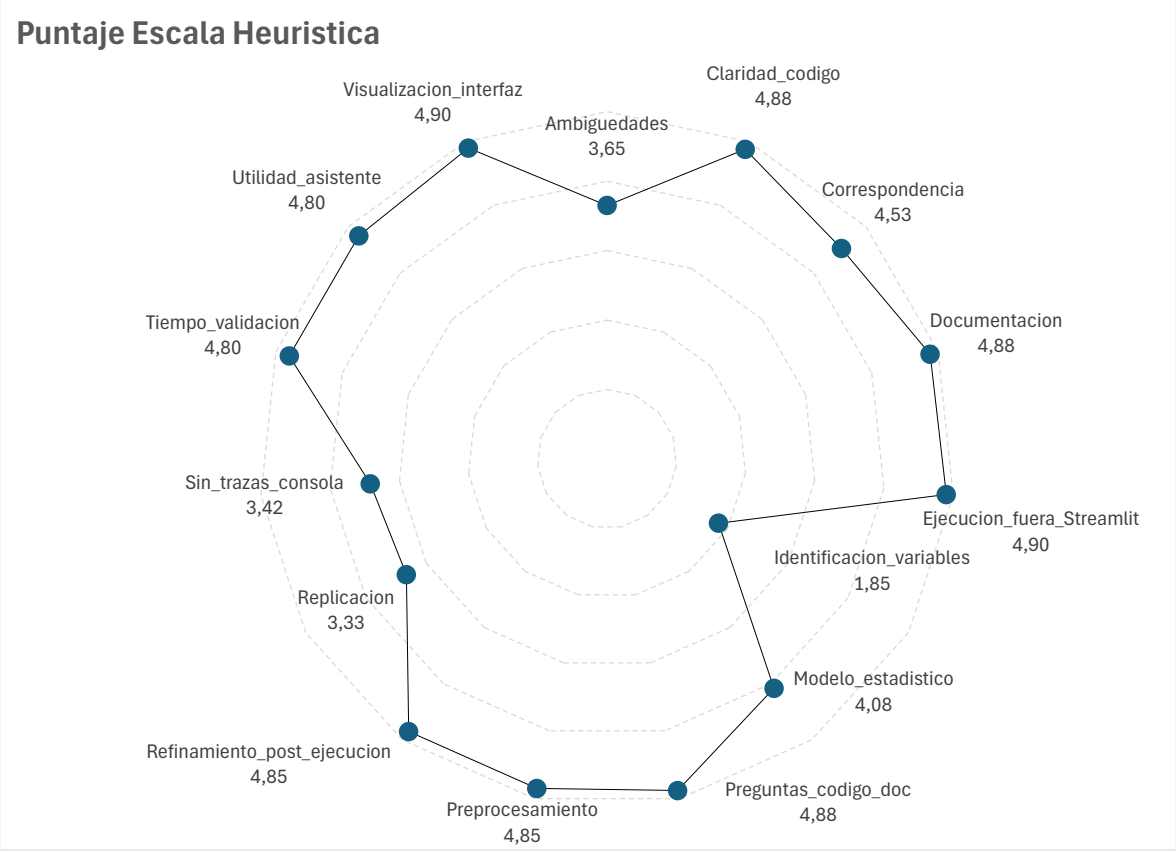


Figura 28 .– Desempeño promedio por criterio en la evaluación heurística funcional del Asistente Multiagente IA

Correspondencia: Correspondencia del código generado con los métodos del artículo; **Replicacion:** Replicación de resultados (tablas, figuras, coeficientes); **Identificacion_variables:** Identificación automática de variables clave; **Modelo_estadistico:** Elección adecuada del modelo estadístico; **Preprocesamiento:** Inclusión de pasos de preprocesamiento; **Claridad_codigo:** Claridad del código generado; **Documentacion:** Documentación técnica generada; **Tiempo_validacion:** Tiempo de evaluación/validación reducido; **Ambiguedades:** Manejo de ambigüedades; **Utilidad_asistente:** Utilidad percibida del asistente para automatizar la validación científica; **Ejecucion_fuera_Streamlit:** El código generado puede ejecutarse y validarse fácilmente fuera del entorno Streamlit; **Sin_trazas_consola:** El sistema no muestra trazas innecesarias por consola; **Refinamiento_post_ejecucion:** El sistema permite continuar con refinamiento del código o documentación tras la ejecución; **Visualizacion_interfaz:** Visualización del código generado en la interfaz; **Preguntas_codigo_doc:** Preguntas sobre el código y documentación

La evaluación funcional del Asistente Multiagente IA, basada en 15 ítems distribuidos entre criterios de validación científica externa y usabilidad interna, reveló un desempeño general sobresaliente, con una media global de 4.38±0.93 puntos de un máximo de 5pts posibles, lo cual sugiere una alta valoración por parte de los evaluadores, con una dispersión relativamente baja en las puntuaciones asignadas. Este resultado indica que, en promedio, el sistema cumple satisfactoriamente con los principios de correspondencia analítica, replicación de resultados, claridad del código, portabilidad, y utilidad percibida para automatizar la validación de artículos científicos. La consistencia en las valoraciones también refleja que tanto la lógica Asistente Multiagente IA como la interfaz fueron interpretadas de forma similar por los distintos evaluadores, consolidando así la robustez del desempeño global del sistema evaluado.

En cuanto a la Funcionalidad Externa – Validación científica, los resultados evidencian un desempeño robusto del Asistente Multiagente IA en los aspectos relacionados con la capacidad de replicación y análisis estadístico. Los ítems más destacados fueron Claridad del código generado, Documentación técnica generada y Utilidad percibida del asistente, todos con una media de 4.88±0.33), lo que refleja una percepción homogénea de excelencia entre los evaluadores. También se obtuvieron puntuaciones altas en Inclusión de pasos de preprocesamiento (media 4.85 ± 0.36) y Tiempo de evaluación/validación reducida (4.80 ± 0.41), lo cual subraya la capacidad del asistente para facilitar la validación científica de manera eficiente. En contraste, se identificaron oportunidades de mejora en ítems como Identificación automática de variables clave, que presentó la puntuación más baja (media 1.85 ± 0.66) dentro de esta dimensión, señalando limitaciones del sistema para reconocer adecuadamente las variables dependientes e independientes en los artículos científicos. Asimismo, la Replicación de resultados obtuvo una media moderada (3.33 ± 0.69), lo que sugiere que, si bien el asistente logra generar código funcional, la fidelidad en la reproducción exacta de tablas, figuras y coeficientes aún requiere ajustes. En general,

esta dimensión muestra un desempeño muy favorable con una media agrupada de 4.09 ± 1.01, indicando que el asistente es funcional y útil para apoyar la validación científica, aunque con áreas específicas que pueden ser optimizadas (Tabla 1 y Figura 28).

Respecto a la Funcionalidad Interna – Usabilidad del asistente, los hallazgos reflejan un nivel de desempeño ampliamente positivo. En primer lugar, se destacan indicadores sobresalientes como Visualización del código generado en la interfaz (4.92 ± 0.27) y Preguntas sobre el código y documentación (4.88 ± 0.33), evidenciando que el asistente no solo muestra los resultados de manera clara y accesible, sino que también permite un proceso de retroalimentación interactivo y eficaz. Asimismo, ítems como El código puede ejecutarse y validarse fácilmente fuera del entorno Streamlit (4.90 ± 0.30) y Refinamiento del código o documentación tras la ejecución (4.85 ± 0.36) resaltan la portabilidad del código y la flexibilidad del flujo de trabajo multiagente, características clave para investigadores que requieren adaptar y mejorar sus análisis de forma continua. Por otro lado, el criterio El sistema no muestra trazas innecesarias por consola obtuvo una media inferior (3.42 ± 0.50), lo que sugiere una oportunidad de mejora en la limpieza del output técnico del sistema. Aun con esta observación, la media general para esta dimensión fue de 4.66 ± 0.51, lo que indica que el asistente presenta una usabilidad altamente satisfactoria y coherente con los principios de diseño centrado en el usuario, permitiendo una interacción fluida y eficaz durante la validación científica (Tabla 1 y Figura 28).

4.3 Evaluación de la Aplicabilidad y Usabilidad del Asistente en Contextos Reales de Investigación Biomédica

En la Tabla 2 describe el perfil sociodemográfico, formativo y de experiencia de los 34 participantes incluidos en el estudio, distribuidos según su nivel de formación académica. La mediana de edad general fue de 36 años (RIC: 29–41), con un incremento progresivo de edad en relación con el nivel educativo: 28 años en especialización, 39 años en doctorado y 48 años en maestría. En cuanto al sexo, predominó la participación masculina (62%), aunque con una distribución desigual entre niveles educativos. Mientras que en el

grupo de especialización la mayoría de los participantes fueron mujeres (64%), en el doctorado el 92% fueron hombres.

Tabla 2.– Perfil de los participantes según nivel de formación académico

Características	Total n = 34[1]	Especialización n = 14[1]	Maestría n = 7[1]	Doctorado n = 13[1]
Edad	36 (29-41)	28 (27-31)	48 (35-58)	39 (36-41)
Sexo				
Femenino	13 (38%)	9 (64%)	3 (43%)	1 (7.7%)
Masculino	21 (62%)	5 (36%)	4 (57%)	12 (92%)
Área de Formación*				
Ciencias biomédicas	28 (82%)	10 (71.4%)	6 (85.7%)	12 (92.3%)
Salud pública / Epidemiología	8 (24%)	5 (35.7%)	2 (28.6%)	1 (7.7%)
Estadística / Bioestadística / Ciencia de Datos	2 (6%)	0 (NA%)	0 (NA%)	2 (15.4%)
Ciencias computacionales	3 (9%)	0 (NA%)	0 (NA%)	3 (23.1%)
Experiencia (años)				
Menor a un año	7 (21%)	7 (50%)	0 (0%)	0 (0%)
1–3 años	10 (18%)	6 (43%)	3 (43%)	1 (7.7%)
4–7 años	6 (18%)	1 (7%)	2 (29%)	3 (23%)
Más de 7 años	11 (32%)	0 (0%)	2 (29%)	9 (69%)

[1] Median (IQR); n (%); * Un participante puede tener mas de un área de formación

Respecto al área de formación, predominaron los profesionales provenientes de las ciencias biomédicas (82%), muchos de ellos especialistas clínicos y subespecialistas. En menor proporción, se incluyeron participantes con formación en salud pública/epidemiología (24%), estadística/bioestadística/ciencia de datos (6%) y ciencias computacionales (9%). En cuanto a la experiencia profesional mostró una distribución heterogénea. El 32% de los participantes refirieron más de 7 años de experiencia laboral, mientras que el 21% reportó menos de un año. Los participantes del nivel de doctorado concentraron los años de mayor experiencia, mientras que en el grupo de especialización predominó la inexperiencia relativa (50% con menos de un año).

Tabla 3.– Nivel de competencia en software estadístico (SPSS, STATA, R y Python) por formación académica

Software	Total n = 34[1]	Especialización n = 14[1]	Maestría n = 7[1]	Doctorado n = 13[1]
SPSS				
Avanzado	2 (5.9%)	1 (7.1%)	0 (0%)	1 (7.7%)
Intermedio	8 (24%)	1 (7.1%)	2 (29%)	5 (38%)
Básico	7 (21%)	2 (14%)	2 (29%)	3 (23%)
Ninguno	17 (50%)	10 (71%)	3 (43%)	4 (31%)
STATA				
Avanzado	1 (2.9%)	0 (0%)	0 (0%)	1 (7.7%)
Intermedio	3 (8.8%)	0 (0%)	0 (0%)	3 (23%)
Básico	9 (26%)	4 (29%)	1 (14%)	4 (31%)
Ninguno	21 (62%)	10 (71%)	6 (86%)	5 (38%)
R				
Avanzado	2 (5.9%)	0 (0%)	0 (0%)	2 (15%)
Intermedio	14 (41%)	1 (7.1%)	0 (0%)	3 (23%)
Básico	4 (12%)	2 (14%)	6 (86%)	6 (46%)
Ninguno	14 (41%)	11 (79%)	1 (14%)	2 (15%)
Python				
Avanzado	1 (2.9%)	0 (0%)	0 (0%)	1 (7.7%)
Intermedio	13 (38%)	0 (0%)	0 (0%)	2 (15%)
Básico	2 (5.9%)	4 (29%)	5 (71%)	5 (38%)
Ninguno	18 (53%)	11 (79%)	2 (29%)	5 (38%)

[1] n (%)

De forma general, se evidencia un bajo dominio avanzado de estos programas, con predominio de niveles básico, intermedio o incluso ausencia total de experiencia. En el caso de SPSS, el 50% de los participantes indicó no tener ningún dominio del software, y solo un 5.9% reportó un nivel avanzado. Entre los niveles académicos, el grupo de especialización presentó el porcentaje más alto de inexperiencia (71%), mientras que en doctorado se observó mayor diversidad en los niveles de competencia, destacándose un 38% en nivel intermedio. Para STATA, los datos revelan aún menor familiaridad: el 62% de los participantes indicó no usarlo y apenas el 2.9% manifestó un dominio avanzado (

Tabla **3**).

En cuanto a R, el panorama cambia ligeramente. Aunque el 41% de los participantes también indicó no tener competencia en su uso, destaca que el 41% declaró un nivel intermedio, el más alto entre todos los softwares evaluados. Y por último Python mostró resultados similares a R: 53% de los participantes indicaron no tener experiencia con esta herramienta, y el 38% declaró un nivel intermedio.

Tabla 4.– Frecuencia de uso de análisis estadístico y experiencia previa con herramientas de inteligencia artificial según nivel académico

Indicadores	**Total** n = 34[1]	**Especialización** n = 14[1]	**Maestría** n = 7[1]	**Doctorado** n = 13[1]
Uso de las Estadísticas				
Siempre	6 (18%)	0 (0%)	0 (0%)	6 (46%)
Frecuentemente	11 (32%)	2 (14%)	4 (57%)	5 (38%)
Ocasionalmente	13 (38%)	8 (57%)	3 (43%)	2 (15%)
Muy poco	4 (12%)	4 (29%)	0 (0%)	0 (0%)
Experiencia con IA/LLMs				
Uso previo de herramientas IA para análisis estadístico	12 (35%)	3 (21%)	2 (29%)	7 (54%)
Uso previo de LLMs (ChatGPT, Gemini, Perplexity, DeepSeek y otros.)	28 (82%)	12 (86%)	5 (71%)	11 (85%)

[1] n (%)

En relación con el uso de la estadística, se observa que solo el 18% de los participantes utiliza análisis estadísticos de forma constante (“siempre”), mientras que el mayor grupo (38%) lo hace “ocasionalmente”. Este patrón se distribuye de forma diferenciada según el nivel de formación: el grupo de doctorado concentra el uso más frecuente (46% “siempre”, 38% “frecuentemente”), lo que sugiere una integración más sistemática de los análisis estadísticos en sus actividades académicas o investigativas Tabla 4.

En cuanto a la experiencia con inteligencia artificial aplicada al análisis estadístico, solo el 35% de los participantes indicó haber utilizado este tipo de herramientas previamente. Al analizarlo por nivel académico, destaca que más de la mitad del grupo doctoral (54%) sí ha tenido contacto con IA, mientras que en especialización y maestría los porcentajes fueron

menores (21% y 29%, respectivamente). Por otro lado, la utilización de Modelos de Lenguaje Grande (LLMs) como ChatGPT, Gemini, Perplexity o DeepSeek fue considerablemente alta en toda la muestra, con un 82% de participantes reportando haber usado alguna de estas herramientas. Esta proporción es similar entre los distintos niveles académicos, lo que demuestra que los LLMs ya están integrados de manera informal o exploratoria en el quehacer académico, incluso cuando los participantes no los relacionan directamente con análisis estadístico.

4.3.1 Evaluación Cuantitativa - System Usability Scale (SUS)

Los usuarios que interactuaron con el sistema proporcionaron una valoración general en términos de facilidad de uso, claridad en la navegación y utilidad percibida. Esta apreciación fue capturada a través de la escala System Usability Scale (SUS), cuyos resultados reflejan el grado de aceptación y satisfacción frente al uso del Asistente Multiagente IA. La evaluación cuantitativa obtenida permitió identificar fortalezas y áreas de mejora, aportando una aproximación objetiva a la experiencia global del usuario.

La evaluación de usabilidad mediante la escala SUS mostró un puntaje total mediano de 78.8 (72.5–87.5). Por nivel académico, el grupo con doctorado obtuvo el mayor puntaje (85.0 [75.0–87.5]), seguido por maestría (77.5 [75.0–85.0]) y especialización (76.3 [60.0–85.0]). En las dimensiones clásicas, la eficacia fue mayor en especialización (8.8 [7.5–10.0]) que en maestría (8.1 [8.1–8.8]) y doctorado (8.1 [6.9–9.4]). La eficiencia se destacó en el grupo doctoral (10.0 [8.3–10.0]) frente a maestría (8.3 [7.5–8.3]) y especialización (7.5 [5.8–9.2]). En satisfacción, el grupo con doctorado reportó mejor percepción (7.5 [6.7–8.3]) comparado con maestría (6.7 [6.7–7.5]) y especialización (5.8 [5.8–7.5]).

Tabla 5.– Resultados de la evaluación a través del System Usability Scale (SUS)

Puntaje SUS	Total n = 34[1]	Especialización N = 14[1]	Maestría N = 7[1]	Doctorado N = 13[1]
Puntaje Total	78.8 (72.5, 87.5)	76.3 (60.0, 85.0)	77.5 (75.0, 85.0)	85.0 (75.0, 87.5)
Dimensiones				
Eficacia	8.8 (7.5, 9.4)	8.8 (7.5, 10.0)	8.1 (8.1, 8.8)	8.1 (6.9, 9.4)
Eficiencia	8.3 (6.7, 10.0)	7.5 (5.8, 9.2)	8.3 (7.5, 8.3)	10.0 (8.3, 10.0)
Satisfacción	6.7 (5.8, 7.5)	5.8 (5.8, 7.5)	6.7 (6.7, 7.5)	7.5 (6.7, 8.3)
Dimensiones Emergentes				
Usabilidad	8.3 (7.1, 9.2)	8.1 (6.7, 9.2)	7.9 (7.5, 8.8)	8.8 (7.9, 9.6)
Aprendizaje	7.5 (6.3, 7.5)	6.3 (5.0, 7.5)	7.5 (6.3, 7.5)	7.5 (6.3, 7.5)
Clasificación				
Pobre	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Aceptable	6 (18%)	5 (36%)	0 (0%)	1 (7.7%)
Buena	14 (41%)	4 (29%)	5 (71%)	5 (38%)
Excelente	14 (41%)	5 (36%)	2 (29%)	7 (54%)

[1] Median (Q1, Q3); n (%)

Las dimensiones emergentes también favorecieron a los más avanzados. En usabilidad, doctorado obtuvo 8.8 (7.9–9.6), especialización 8.1 (6.7–9.2) y maestría 7.9 (7.5–8.8). En aprendizaje, doctorado y maestría compartieron 7.5 (6.3–7.5), mientras especialización tuvo 6.3 (5.0–7.5). En cuanto a la clasificación SUS, ningún participante evaluó el sistema como “pobre”. En total, el 82% lo consideró “bueno” (41%) o “excelente” (41%). Por nivel, la categoría “excelente” fue reportada por 54% del doctorado, 29% de maestría, y 36% de especialización.

Al relacionar esta clasificación SUS con los años de experiencia, se encontró una asociación estadísticamente significativa entre la experiencia laboral y la percepción de usabilidad del sistema ($p = 0.04462$, prueba exacta de Fisher). En particular, los participantes con más de 7 años de experiencia concentraron las evaluaciones más altas: el 64% (7 de 11) calificaron la usabilidad como “excelente”. En contraste, quienes reportaron ninguna experiencia previa o menos de 4 años tendieron a evaluarlo como “aceptable” o “bueno”. Este hallazgo sugiere que la percepción positiva del Asistente Multiagente IA no solo está influenciada por el nivel académico, sino también por la trayectoria profesional y el grado de exposición a herramientas tecnológicas y procesos de análisis de datos.

4.3.2 Análisis Semántico-Emocional de la Percepción del Usuario

Más allá de la valoración cuantitativa, los testimonios y expresiones lingüísticas de los usuarios revelaron matices emocionales que enriquecieron la comprensión de su experiencia con el Asistente Multiagente IA. A través del análisis semántico-emocional, se identificó no solo qué piensan los usuarios, sino también cómo lo sienten. Este enfoque permitió mapear emociones como confianza, alegría, sorpresa o frustración, y asociarlas a elementos clave del sistema como la utilidad, la claridad, la eficacia y la interacción. Obteniéndose redes visuales que ofrecieron una mirada profunda y estructurada sobre la dimensión afectiva de la experiencia, ampliando la interpretación de los resultados más humanizada.

4.3.2.1 Análisis Semántico-Emocional de la Percepción del Usuario – Eficacia

En esta red (Figura 29), se observa un predominio de emociones positivas, especialmente *positivo*, *confianza*, *alegría* y *anticipación*, que agrupan términos como *logro*, *formación*, *eficaz*, *conocimiento*, *asistente*, *objetivo*, *investigador* y *práctico*. Estas asociaciones refuerzan la percepción favorable de los usuarios frente a la eficacia del sistema en el cumplimiento de sus propósitos, en concordancia con los resultados obtenidos mediante la escala System Usability Scale (SUS).

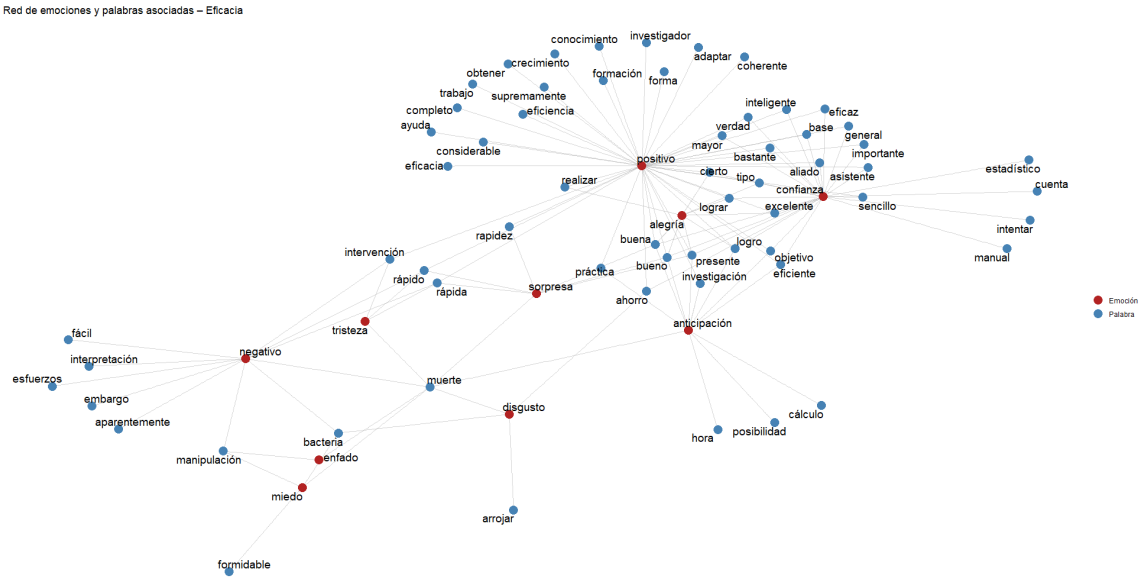


Figura 29.- Red semántico-emocional de la dimensión «Eficacia».

Por otra parte, aunque emociones negativas como *disgusto*, *miedo*, *enfado* y *tristeza* están presentes, su densidad de conexión es menor y se concentran en palabras asociadas a dificultades puntuales como *manipulación*, *bacteria*, *complejidad*, *muerte* y *esfuerzos*, lo cual permite orientar mejoras específicas. Esta coexistencia entre valoraciones positivas dominantes y críticas puntuales refuerza el carácter equilibrado del análisis emocional generado por el Asistente Multiagente IA.

4.3.2.2 Análisis Semántico-Emocional de la Percepción del Usuario – Eficiencia

En esta red (Figura 30), se identifican emociones predominantes como *positivo*, *confianza*, *alegría* y *anticipación*, conectadas con términos como *preparación*, *eficaz*, *rápido*, *solución*, *importante*, *recursos* y *conocimiento*. Estas relaciones sugieren que los usuarios perciben el sistema como funcional, rápido y útil, lo que refuerza la dimensión de eficiencia evaluada previamente por medios cuantitativos (SUS).

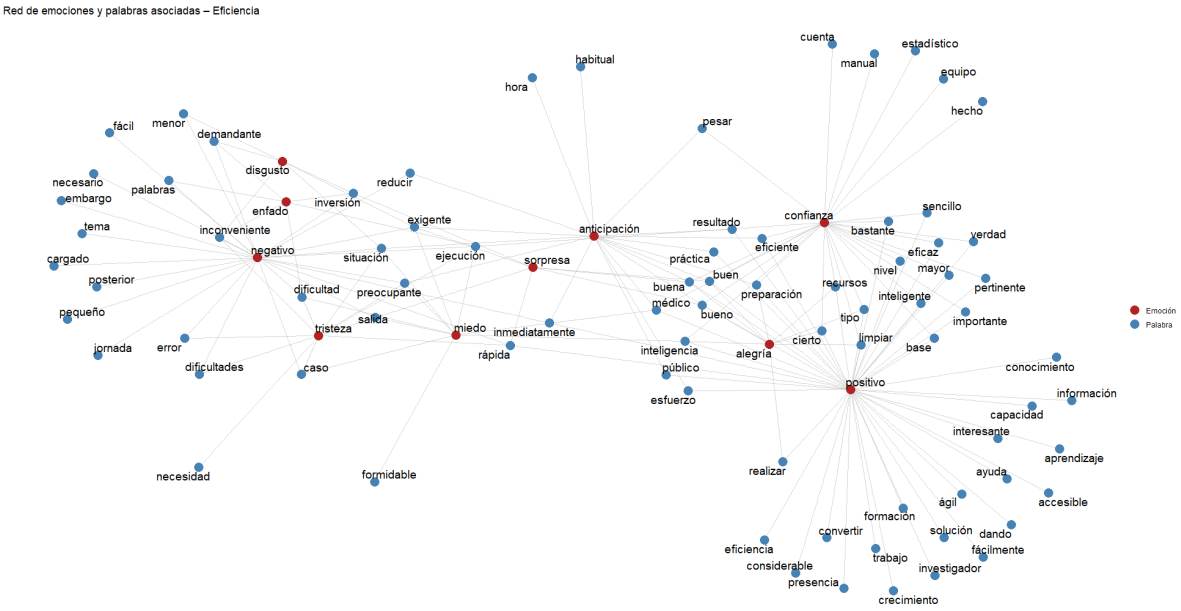


Figura 30.- Red semántico-emocional de la dimensión «Eficiencia» .

En contraste, emociones negativas como *disgusto*, *enfado*, *tristeza* y *miedo* se vinculan a palabras como *problema*, *inconveniente*, *carga* y *exigente*, reflejando situaciones que, aunque menos frecuentes, podrían afectar la percepción de eficiencia en determinados escenarios. Esta distribución evidencia un dominio de valoraciones positivas, a la vez que señala oportunidades para optimizar la experiencia del usuario.

4.3.2.3 Análisis Semántico-Emocional de la Percepción del Usuario – Satisfacción

En esta red (Figura 31), sobresalen emociones como *positivo*, *confianza*, *alegría* y *sorpresa*, que se relacionan con palabras como *preparación*, *información*, *resultado*, *mejora*, *práctica* y *aprendizaje*. Estas asociaciones reflejan una percepción general de satisfacción con el sistema, especialmente en aspectos vinculados a la utilidad, claridad y aportes a la experiencia investigativa.

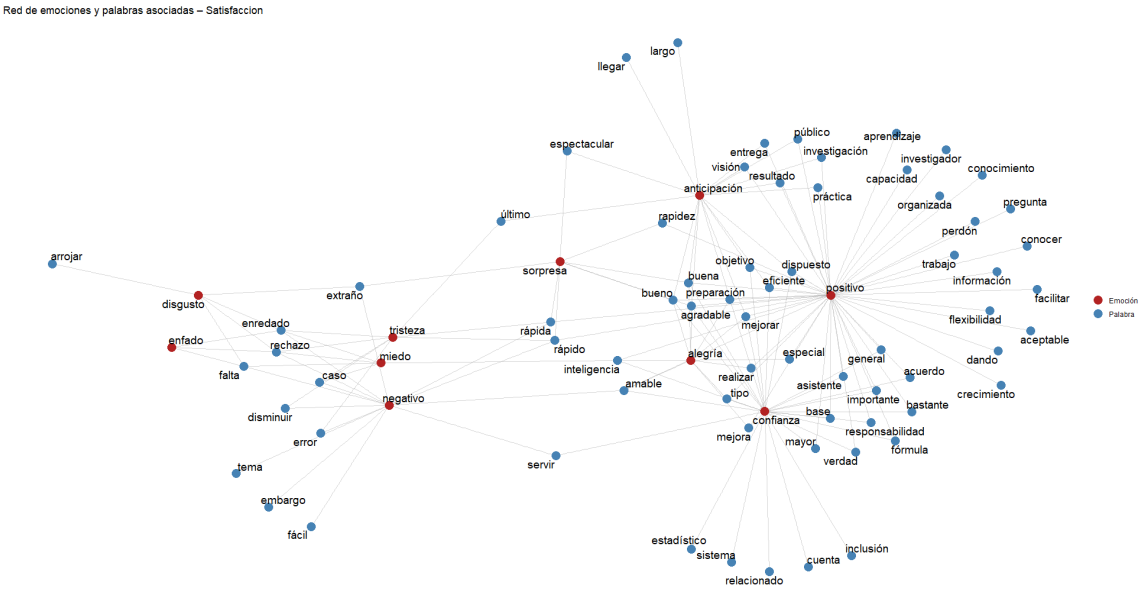


Figura 31.- Red semántico-emocional de la dimensión “Satisfacción”.

Si bien aparecen emociones como *disgusto*, *tristeza* y *miedo*, estas se relacionan con términos como *rechazo*, *error*, *falta* y *enredo*, lo que sugiere aspectos específicos que generaron disconformidad. Sin embargo, su menor grado de conexión en la red indica que se trata de experiencias aisladas, en contraste con una tendencia positiva ampliamente predominante.

4.3.2.4 Valoración Global de Asistente Multiagente

En esta red (Figura 32), que integra las tres dimensiones evaluadas (Eficacia, Eficiencia y Satisfacción), se observa un marcado predominio de emociones positivas, especialmente positivo, confianza, alegría y anticipación. Estas emociones se encuentran fuertemente conectadas a términos como preparación, eficaz, conocimiento, objetivo, crecimiento, formación, resultado y asistente, lo cual evidencia una percepción favorable consolidada del sistema en su conjunto. La concentración de asociaciones positivas en torno a conceptos clave refuerza la utilidad percibida, la aplicabilidad práctica y la experiencia satisfactoria durante el uso del sistema.

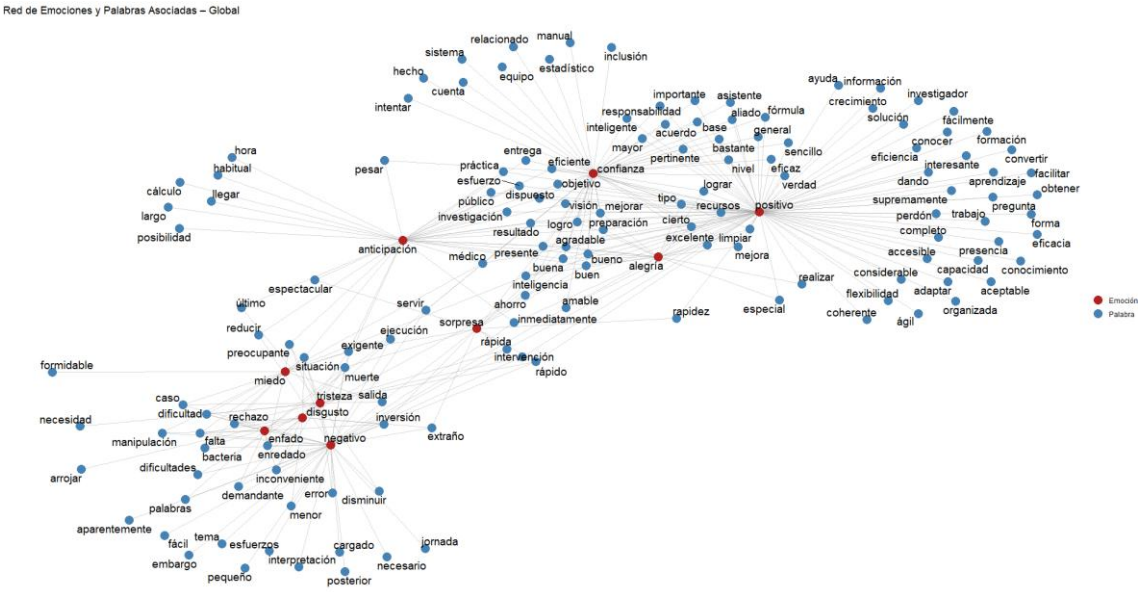


Figura 32.- Red semántico-emocional Global.

Por su parte, las emociones negativas como disgusto, tristeza, miedo y enfado presentan menor centralidad y densidad de conexión. Estas se asocian con términos como error, problema, inconveniente, muerte y rechazo, reflejando experiencias puntuales o dificultades percibidas que, si bien no dominan el discurso emocional, constituyen oportunidades de mejora en aspectos operativos o de accesibilidad. La coexistencia de ambas cargas emocionales, con un claro predominio de lo positivo, confirma la eficacia del Asistente Multiagente IA en la representación estructurada del discurso del usuario, y valida la aceptabilidad del sistema desde una perspectiva global.

4.3.3 Evaluación en la Contribución a la Co-creación de Conocimiento Científico

Al analiza el valor epistémico del uso del Asistente Interactivo Multiagente basado en Modelos de Lenguaje Grande, a través del análisis de un conjunto de datos abiertos del sistema de vigilancia epidemiológica de COVID-19 en Colombia, se demuestra empíricamente cómo un sistema interactivo multiagente, diseñado para ejecutar análisis estadísticos, puede integrarse de forma significativa en los procesos de investigación, más allá de la simple automatización.

Cuadro 10.- Instrucción semántica inicial para la para la co-creación con el Asistente Interactivo Multiagente LLM de flujo de trabajo analítico en R

Enter your coding or scientific problem:

Realiza un análisis estadístico en R a partir de un dataset de casos de COVID-19 en Colombia que contiene fechas ya limpias en formato Date. Crea las variables dias_diagnostico, dias_recuperacion y dias_muerte como la diferencia entre fechas correspondientes. Luego, realiza un análisis descriptivo y analítico cruzado por sexo: incluye resumen estadístico de edad por sexo (media, mediana, min, max), histogramas y boxplots de edad, y tablas de frecuencia de departamento_nom, per_etn_ y nom_grupo_ desagregadas por sexo. Analiza los tiempos clínicos (dias_diagnostico, dias_recuperacion, dias_muerte) con estadísticos y gráficos comparativos por sexo. Adicionalmente, genera tablas cruzadas por sexo para las variables estado, ubicacion, tipo_recuperacion y recuperado. Finalmente, grafica la evolución semanal de casos (fecha_diagnostico), recuperaciones (fecha_recuperado) y fallecimientos (fecha_muerte) por sexo. Visualizar todas las tablas en RStudio.

Este es un ejemplo de las primeras filas de la base de datos: ANEXO Base de datos descargada

El Cuadro 10 presenta la instrucción semántica inicial proporcionada al Asistente Interactivo Multiagente, a partir de la cual se estructuró un flujo de análisis estadístico completo en R, aplicado a datos reales de COVID-19. Esta directriz permitió la generación automatizada de variables clínicas temporales, análisis descriptivos desagregados por sexo, tabulaciones cruzadas y visualizaciones de evolución temporal. La correcta interpretación y ejecución de dichas tareas por parte del sistema evidencia su capacidad para intervenir de manera significativa en procesos de análisis científico, aportando así un valor epistémico tangible en contextos de cocreación humano-máquina.

Cuadro 11.– Descomposición funcional de tareas por el Agente Planificador-Analizador

Planned Subtasks:

1. Cargar el dataset en R y asegurar que las variables de fecha estén en formato Date.
2. Crear las variables dias_diagnostico (fecha_diagnostico - fecha_inicio_sintomas), dias_recuperacion (fecha_recuperado - fecha_diagnostico), y dias_muerte (fecha_muerte - fecha_diagnostico), manejando valores faltantes según corresponda.
3. Realizar un resumen estadístico de la edad por sexo (media, mediana, mínimo, máximo) y generar histogramas y boxplots de edad por sexo.
4. Generar tablas de frecuencia de departamento_nom, per_etn_, y nom_grupo_ desagregadas por sexo.
5. Analizar los tiempos clínicos (dias_diagnostico, dias_recuperacion, dias_muerte) con estadísticos descriptivos y gráficos comparativos por sexo.
6. Crear tablas cruzadas por sexo para las variables estado, ubicacion, tipo_recuperacion y recuperado.
7. Graficar la evolución semanal de casos (fecha_diagnostico), recuperaciones (fecha_recuperado) y fallecimientos (fecha_muerte) por sexo.
8. Visualizar todas las tablas y gráficos en RStudio.

El Cuadro 11 presenta la descomposición funcional de tareas generada por el Agente Planificador-Analizador a partir de la instrucción semántica inicial. Esta estructuración automatizada del flujo de trabajo permitió una secuenciación lógica y detallada de las operaciones estadísticas requeridas, desde la preparación de datos y creación de variables temporales clínicas, hasta la generación de resúmenes descriptivos, tabulaciones cruzadas por sexo y visualización de tendencias semanales. La capacidad del sistema para desglosar y planificar de manera coherente estas acciones evidencia su utilidad como herramienta de soporte analítico, facilitando procesos de co-creación científica estructurada en entornos de análisis de datos biomédicos.

Cuadro 12.– Heurística inferencial del Agente Planificador: estructura de datos, transformaciones y flujos de análisis

Research Summary:

Pensamientos:

- Cargar el dataset en R y asegurar que las variables de fecha estén en formato Date. \
- Crear las variables dias_diagnostico (fecha_diagnostico - fecha_inicio_sintomas), dias_recuperacion (fecha_recuperado - fecha_diagnostico), y dias_muerte (fecha_muerte - fecha_diagnostico), manejando valores faltantes según corresponda. \
- Realizar un resumen estadístico de la edad por sexo (media, mediana, mínimo, máximo) y generar histogramas y boxplots de edad por sexo. \
- Generar tablas de frecuencia de departamento_nom, per_etn_, y nom_grupo_ desagregadas por sexo. \
- Analizar los tiempos clínicos (dias_diagnostico, dias_recuperacion, dias_muerte) con estadísticos descriptivos y gráficos comparativos por sexo. \
- Crear tablas cruzadas por sexo para las variables estado, ubicacion, tipo_recuperacion y recuperado. \
- Graficar la evolución semanal de casos (fecha_diagnostico), recuperaciones (fecha_recuperado) y fallecimientos (fecha_muerte) por sexo. \
- Visualizar todas las tablas y gráficos en RStudio.

El Cuadro 12 sintetiza la heurística inferencial generada por el Agente Planificador, en la cual se establecen las operaciones clave requeridas para estructurar, transformar y analizar el conjunto de datos en R. Esta representación no solo refleja la secuencia lógica de los pasos analíticos, sino que también evidencia la capacidad del agente para generar un flujo de razonamiento que articula la preparación de datos, la creación de variables derivadas, el análisis comparativo por sexo y la visualización de patrones temporales. La claridad de esta heurística resalta la función del sistema como mediador entre la intención investigativa humana y su operacionalización computacional, reafirmando su rol en la cocreación de conocimiento científico reproducible y estructurado.

Cuadro 13.– Representación esquemática del plan de investigación por el Agente Investigador

Pensamientos:

1. El plan es claro y cubre una exploración descriptiva y analítica integral del conjunto de datos de COVID-19 en R, centrándose en intervalos de tiempo, datos demográficos y resultados clínicos, todos desglosados por sexo.
2. La información clave necesaria incluye: definiciones de variables, manejo de datos faltantes, métodos estadísticos y de visualización apropiados y posibles desafíos de datos (por ejemplo, valores faltantes o inconsistentes en fechas, variables categóricas).
3. Los principales desafíos probablemente sean los valores de fecha faltantes o inconsistentes, garantizar el cálculo correcto de los intervalos de tiempo y manejar variables categóricas con posibles errores tipográficos o inconsistencias.
4. Los paquetes R relevantes incluyen: dplyr, lubridate, ggplot2 y posiblemente funciones de tabla como table(), prop.table() y tabyl() de janitor.
5. El resumen debe proporcionar tanto los pasos metodológicos como las consideraciones prácticas para cada punto de análisis.

Resumen de la investigación:

- Punto A: Preparación del conjunto de datos
  - Asegúrese de que todas las variables de fecha (fecha_inicio_sintomas, fecha_diagnostico, fecha_recuperado, fecha_muerte) estén en formato de fecha utilizando as.Date().
  - Verifique y gestione adecuadamente las fechas faltantes o mal formateadas, ya que esto afectará los cálculos del intervalo de tiempo.
  - Utilice dplyr para manipular y limpiar datos.
- Punto B: […] continua en Anexo: Continuación del Cuadro 13.– Representación esquemática del plan de investigación por el Agente Investigador

El Cuadro 13 muestra la representación esquemática del plan de investigación inferido por el Agente-Investigador, en el que se articula una exploración descriptiva y analítica del conjunto de datos de COVID-19. Esta heurística integra definiciones conceptuales, identificación de desafíos metodológicos —como la gestión de fechas faltantes y variables categóricas inconsistentes—, y la recomendación explícita de paquetes y funciones en R apropiados para cada etapa del análisis. La claridad de este esquema no solo guía el flujo operativo del análisis, sino que refleja la capacidad del sistema para traducir objetivos científicos en una estrategia técnica detallada y reproducible, lo que refuerza su aporte a la cocreación estructurada del conocimiento científico.

Cuadro 14.– Inicialización automatizada del entorno computacional por el Agente Codificador

```
R:

# ==== Optimized Epidemiological Data Analysis Script ====

# ---- 0. Utility Functions and Package Checks ----
ensure_packages <- function(pkgs) {
  new_pkgs <- pkgs[!(pkgs %in% installed.packages()[,"Package"])]
  if(length(new_pkgs)) {
    install.packages(new_pkgs)
    message("Installed missing packages: ", paste(new_pkgs, collapse = ", "))
  }
  invisible(lapply(pkgs, library, character.only = TRUE))
}

ensure_packages(c(
  "readr", "dplyr", "lubridate", "ggplot2", "DT",
  "tidyr", "scales", "gtsummary", "rlang", "utils"
))

[…] Continua en Anexo: Continuación del Cuadro 14.- Inicialización automatizada del
entorno computacional por el Agente Codificador
```

El Cuadro 14 ilustra la rutina de inicialización automatizada del entorno computacional generada por el Agente-Codificador. A través de una función de verificación e instalación condicional de paquetes (`ensure_packages`), el sistema garantiza la disponibilidad de bibliotecas necesarias para el análisis estadístico, incluyendo `dplyr`, `lubridate`, `ggplot2`, `DT`, `readr`, entre otras. Esta función, diseñada para ejecutarse de forma reproducible y autónoma, permite configurar dinámicamente el entorno de trabajo en R, minimizando errores por paquetes faltantes y estandarizando la infraestructura computacional del análisis. Este procedimiento refleja una estrategia orientada a la robustez y portabilidad del sistema en escenarios reales de investigación científica.

Cuadro 15.– Evaluación de la calidad metodológica del script por el Agente Documentador

Documentación y preguntas y respuestas

Pensamientos:

1. El código es un script R robusto y modular para el análisis epidemiológico de datos de casos de COVID-19, con sólidas funciones de validación, transformación y generación de informes de datos.
2. Su diseño enfatiza el manejo de errores, la reproducibilidad y la salida clara, utilizando convenciones tidyverse y paquetes R modernos.
3. Las funciones clave realizan estadísticas de resumen, visualización, tabulación cruzada y análisis de series de tiempo por sexo, con todos los resultados mostrados de forma segura a través de tablas o gráficos.
4. El script es adecuado para conjuntos de datos con calidad variable, ya que anticipa y maneja datos faltantes e inconsistencias.
5. El uso de ecuaciones LaTeX es principalmente relevante para describir el cálculo de intervalos de tiempo clínicos.

El Cuadro 15 presenta la evaluación metodológica realizada por el Agente Documentador sobre el script generado para el análisis de datos de COVID-19. Se destaca la robustez y modularidad del código, su capacidad para realizar validaciones, transformar datos y generar resultados reproducibles mediante convenciones del ecosistema `tidyverse`. El diseño del script prioriza el manejo eficiente de errores, la claridad de salida y la interoperabilidad con formatos de presentación (incluyendo ecuaciones LaTeX para intervalos de tiempo clínico). Asimismo, se resalta su aplicabilidad en contextos con datos incompletos o heterogéneos, consolidando su utilidad en análisis epidemiológicos automatizados con soporte en IA.

Cuadro 16.– Documentación estructural inicial del script: objetivos, funciones y estructura funcional

Descripción general
Este script de R realiza un análisis estadístico exhaustivo de los datos de casos de COVID-19 en Colombia, centrándose en las cronologías clínicas y los patrones demográficos por sexo. El flujo de trabajo incluye:

- **Carga y validación de datos robustas** : garantiza que las columnas y los archivos necesarios estén presentes.
- **Transformación de datos** : calcula variables de tiempo clínico clave (por ejemplo, tiempo hasta el diagnóstico, recuperación, muerte).
- **Estadística descriptiva** : Resume y visualiza la edad y los tiempos clínicos por sexo.
- **Análisis categóricos** : genera tablas de frecuencia y tabulaciones cruzadas para variables demográficas y clínicas importantes.
- **Análisis de series de tiempo** : sigue la evolución semanal de casos, recuperaciones y muertes por sexo.
- **Manejo y generación de informes de errores** : utiliza funciones auxiliares para gestionar datos vacíos, columnas faltantes y errores de trazado, lo que garantiza una ejecución fluida incluso con datos incompletos.

El script aprovecha los siguientes paquetes principales: **dplyr** , **ggplot2** , **lubridate** , **readr** , **DT** , **tidyr** , **scales** y **gtsummary** .

Funciones
0. Funciones de utilidad y comprobaciones de paquetes
ensure_packages(pkgs)
Se asegura de que todos los paquetes necesarios estén instalados y cargados. Instala los paquetes faltantes y carga todos los paquetes listados mediante [nombre del archivo] library().

- **Entrada:** Vector de caracteres de nombres de paquetes.
- **Salida:** Ninguna (efectos secundarios: paquetes instalados y cargados).

1. Comprobaciones de existencia de archivos y columnas

- **Comprueba la presencia del archivo de datos de entrada** (documentos.csv).
- **Valida que todas las columnas requeridas** estén presentes en el conjunto de datos después de la carga.

[…] Continua en el Anexo: Continuación del Cuadro 16.– Documentación estructural inicial del script: objetivos, funciones y estructura funcional

El Cuadro 16 sintetiza la estructura funcional del script desarrollado por el sistema multiagente, destacando sus objetivos, funciones clave y lógica operativa. El script fue diseñado para realizar un análisis estadístico exhaustivo de datos clínicos y demográficos de casos de COVID-19 en Colombia, abarcando tareas como la validación robusta de archivos, la transformación de fechas en variables temporales clínicas, el análisis descriptivo y categórico por sexo, y el seguimiento de series de tiempo. Además, incluye rutinas para la detección y manejo automatizado de errores, lo que permite su ejecución incluso en condiciones de datos incompletos. La documentación evidencia una arquitectura

modular, con funciones específicas para verificación de paquetes, control de columnas requeridas y preparación del entorno, reforzando su reproducibilidad y aplicabilidad en escenarios reales de investigación epidemiológica.

Todas las tablas y gráficos producto del script automatizado generado por el sistema multiagente se presentan en el Anexo: Resultados gráficos y tabulares generados por el Asistente Multiagente IA en R, organizados por tipo de análisis y desagregados por sexo, según lo establecido en el flujo de trabajo definido por el Agente Planificador y Documentador.

Capítulo 5: Discusiones

El sistema opera como un marco colaborativo donde distintos agentes inteligentes, cada uno con un rol especializado, trabajan en conjunto con un usuario humano para descomponer problemas complejos, investigar soluciones, generar código, verificar su corrección, optimizarlo y documentarlo. La interactividad es clave, permitiendo que el usuario no solo inicie el proceso, sino que también guíe, valide y refine las salidas en cada fase, asegurando que el producto final se alinee precisamente con sus requisitos y expectativas. Este enfoque contrasta marcadamente con los modelos de generación de código puramente autónomos, que a menudo carecen de la capacidad de auto-corrección o de la comprensión profunda de las sutilezas del dominio, al integrar la intuición, el juicio y el conocimiento de dominio humano en un bucle de retroalimentación continuo y dinámico. La motivación subyacente es superar las limitaciones inherentes a los LLM monolíticos en tareas complejas de ingeniería de software, donde la precisión, la robustez y la adherencia a especificaciones no triviales son primordiales.

La automatización del análisis estadístico en investigaciones biomédicas ha sido históricamente limitada por la necesidad de competencias técnicas avanzadas, la complejidad de los entornos computacionales y la baja replicabilidad de los flujos de trabajo analíticos. Frente a estas limitaciones, la incorporación de modelos de lenguaje grande (LLMs) como motores de generación de código y documentación representa una solución innovadora con alto potencial transformador. En este estudio se desarrolló un Asistente Interactivo Multiagente, estructurado como un sistema colaborativo compuesto por siete agentes especializados (Planificador, Analizador, Investigador, Codificador, Verificador, Depurador y Documentador), cuya lógica de diseño se basa en el principio de descomposición funcional y cooperación entre módulos autónomos, inspirado en marcos de sistemas multiagente (MAS) aplicados a entornos complejos [121–123].

Cada agente fue diseñado con system prompts personalizados que definen de forma explícita su rol, estructura de salida y expectativas funcionales, una estrategia que permite evitar ambigüedades en la generación de respuestas y fortalece la interoperabilidad del sistema. La literatura ha mostrado que la ingeniería de prompts orientada a tareas mejora significativamente la precisión y el control de los LLMs en aplicaciones técnicas como programación, razonamiento lógico y redacción científica [124,125]. El ser modular, además de permitir una implementación progresiva y escalable, facilita el mantenimiento y la mejora por componentes, sin comprometer la integridad funcional del sistema general.

Uno de los aspectos más innovadores del desarrollo fue la adopción de un flujo de trabajo iterativo y no lineal, donde los agentes interactúan con el usuario de manera continua en todas las fases del análisis: desde la planificación del abordaje estadístico hasta la documentación y validación final del código. Esta dinámica rompe con los modelos clásicos de automatización cerrada, donde el sistema actúa de forma autónoma y sin supervisión. En su lugar, se implementó un enfoque Human-in-the-Loop (HITL), en el cual el investigador orienta, supervisa y ajusta cada decisión del sistema, asegurando no solo precisión técnica sino también alineación contextual con los objetivos del estudio [126–128]. Esta integración humana es particularmente relevante en entornos como la biomedicina, donde la variabilidad de los diseños experimentales y la sensibilidad ética de los datos exigen altos niveles de interpretación contextual y control experto.

El sistema se nutre de un componente adicional de valor: una base de datos vectorial FAISS que actúa como sistema de recuperación semántica para acceder a artículos científicos, manuales de software estadístico, guías metodológicas y documentos cargados por el usuario. Esta integración, propia de los sistemas de retrieval-augmented generation (RAG), permite que los agentes utilicen conocimiento externo actualizado y específico del dominio para fundamentar sus decisiones y respuestas, mejorando la relevancia del contenido generado [110,111]. Este mecanismo no solo incrementa la capacidad del sistema para generar análisis estadísticos robustos, sino que además actúa como barrera contra las conocidas "alucinaciones" de los LLMs, al anclar las salidas en fuentes confiables y verificables.

Desde una perspectiva operativa, el enfoque HITL implementado en este asistente promueve una colaboración simétrica entre el humano y la IA, donde la máquina no reemplaza, sino potencia al investigador. Esta idea es coherente con la visión de la medicina aumentada propuesta por Topol (2019) [129].quien destaca que los sistemas de IA deben diseñarse para amplificar la inteligencia humana, no para suplantarla. La capacidad del asistente para incorporar entradas contextuales, corregirse tras la validación humana y adaptar su lógica a las condiciones del problema, le confiere una plasticidad funcional poco habitual en sistemas tradicionales, que operan sobre reglas codificadas sin aprendizaje adaptativo o delimitadas por scripts predefinidos.

Desde el punto de vista técnico, este desarrollo integra diversos avances recientes en inteligencia artificial aplicada: modularidad arquitectónica, prompts orientados a tareas, recuperación semántica y supervisión humana integrada. Este tipo de innovación se alinea con las recomendaciones internacionales para el diseño responsable de sistemas de IA en investigación científica, que destacan la importancia de la transparencia, reproducibilidad, colaboración y actualización constante [129,130].

Desde el punto de vista cuantitativo, los resultados obtenidos revelan un alto grado de consenso entre los evaluadores. El índice de concordancia AC1 (0.831 ± 0.075) evidencia la solidez del instrumento de evaluación y la homogeneidad en la interpretación de los criterios funcionales. Esta métrica resulta más robusta que el clásico coeficiente Kappa en contextos de alta prevalencia o asimetría de respuestas, lo cual la convierte en una herramienta especialmente útil para en evaluaciones de concordancia Inter-evaluador para sistemas complejos [131]. Estos resultados se alinean con investigaciones recientes que han propuesto marcos de validación participativos para agentes inteligentes en salud, donde la concordancia interevaluador es un indicador de transparencia funcional [132,133].

En la dimensión de funcionalidad externa, destacan valores elevados en aspectos clave como la claridad del código (4.88), la documentación técnica generada (4.88), el preprocesamiento automatizado (4.85) y el tiempo de validación (4.80), con niveles de concordancia igualmente altos (AC1 > 0.80). Estos resultados son consistentes con estudios que han evaluado el uso de LLMs en programación asistida o análisis automatizado, donde la claridad del output y la capacidad de seguimiento del flujo lógico

son esenciales para la confiabilidad científica [134,135]. La adecuada documentación del código —frecuentemente ausente en procesos automatizados clásicos— representa un valor agregado, pues favorece la reproducibilidad, la trazabilidad de decisiones analíticas y el entendimiento por parte de otros investigadores, aspectos que la literatura ha identificado como pilares de la ciencia abierta [136–138].

No obstante, la validación también permitió detectar áreas críticas de mejora. En particular, la puntuación baja obtenida por la capacidad del sistema para identificar automáticamente las variables clave (1.85 ± 0.66) evidencia una de las limitaciones actuales de los LLMs en contextos donde se requiere razonamiento estructurado sobre metadatos o diccionarios de variables. Aunque los modelos de lenguaje han demostrado capacidades impresionantes en tareas generales de comprensión de texto, su precisión disminuye cuando deben actuar sobre conjuntos de datos estructurados sin contexto semántico explícito [139,140]. Esta debilidad sugiere la necesidad de enriquecer el sistema con mecanismos de análisis semántico supervisado o estrategias de prompt chaining que permitan establecer relaciones explícitas entre las variables disponibles y los objetivos del análisis. Alternativamente, la integración con ontologías biomédicas como UMLS o MeSH podría facilitar una interpretación más precisa de las entidades estadísticas relevantes [41,141].

Por otra parte, el rendimiento en la dimensión de funcionalidad interna o usabilidad del sistema fue notoriamente alto, con puntuaciones sobresalientes en visualización del código (4.92), capacidad de ejecución externa (4.90) y posibilidad de interacción por preguntas y refinamientos posteriores (4.85). Estos hallazgos reflejan el éxito de una lógica de diseño centrada en el usuario, propia del paradigma Human-Centered AI, que enfatiza la importancia de interfaces interpretables, adaptativas y con retroalimentación en tiempo real durante el proceso de iteración analítica [126,142]. A diferencia de otros asistentes estadísticos autónomos o cerrados, el sistema propuesto no solo permite modificar el código generado, sino también realizar ajustes iterativos, visualizar los resultados y establecer un canal bidireccional de interacción, lo cual incrementa su aceptación y utilidad práctica.

Incluso el ítem con menor puntuación —relativo a la ausencia de trazas innecesarias en consola (3.42 ± 0.50)— mantiene un desempeño aceptable, y representa un aspecto

técnico solucionable mediante filtros de salida o limpieza de logs en la generación de scripts. Este tipo de hallazgo es valioso no por su severidad, sino por su especificidad y utilidad para la mejora incremental y orientado a la mejora incremental del sistema. Como sugieren estudios sobre usabilidad de herramientas basadas en LLMs en programación asistida, la limpieza del entorno de salida y la organización visual del código influyen significativamente en la percepción de calidad del sistema, especialmente entre usuarios con menor experticia técnica [143–145].

Es importante resaltar que todo este proceso de validación se enmarca en un modelo de supervisión cognitiva Human-in-the-Loop (HITL), donde el rol del usuario es central para la verificación, corrección y contextualización del análisis. Esta estrategia ha sido reconocida por su capacidad de aumentar la precisión y reducir el sesgo en sistemas de IA, especialmente en tareas científicas donde la experticia humana sigue siendo insustituible [127,128]. En este contexto, el asistente no reemplaza al investigador, sino que se convierte en un colaborador inteligente que acelera procesos, estandariza procedimientos y potencia el pensamiento analítico humano.

Una de las fortalezas de esta evaluación fue la inclusión de 34 investigadores biomédicos con distintos niveles de formación (especialización, maestría y doctorado) y diversidad en experiencia laboral. Esta heterogeneidad permitió capturar con mayor precisión la aplicabilidad del sistema en contextos reales, donde los usuarios pueden tener desde formación clínica básica hasta competencias estadísticas avanzadas. El 82% pertenecía a áreas biomédicas, mientras que solo un 6% tenía formación en estadística y 9% en ciencias computacionales, lo cual refleja un perfil realista del investigador clínico promedio, que enfrenta análisis complejos sin necesariamente tener formación técnica especializada [136,146].

Este contexto acentúa el valor del asistente desarrollado: su capacidad para facilitar el análisis estadístico en usuarios con escaso dominio de herramientas como R, Python o STATA —programas en los que más del 50% de los participantes declararon no tener experiencia— posiciona al sistema como un puente efectivo entre la necesidad analítica y la capacidad técnica real. Este fenómeno ha sido descrito por autores como Sudha et al. (2023) y Köpf et al. (2023) [147,148], quienes destacan que los LLMs y sistemas

conversacionales pueden democratizar el acceso al análisis de datos al reducir la carga técnica y cognitiva, permitiendo que investigadores sin formación en programación generen productos científicos reproducibles y técnicamente sólidos.

La evaluación mediante la escala estandarizada System Usability Scale (SUS) evidenció una mediana general de 78.8, con una clasificación global de “bueno” a “excelente” en el 82% de los participantes. Este nivel de aceptación es particularmente significativo si se considera la baja competencia técnica inicial en muchos usuarios, y demuestra que el diseño modular, el lenguaje natural de interacción y el control humano de cada paso del proceso —características del enfoque HITL— facilitaron el uso intuitivo del sistema [127].

Los resultados por nivel académico reforzaron esta tendencia: los usuarios con formación doctoral alcanzaron puntuaciones más altas en eficiencia (10.0), satisfacción (7.5), y aprendizaje (7.5), lo que sugiere que los investigadores más experimentados logran aprovechar de manera más eficaz las funcionalidades del asistente. Estudios previos han demostrado que la experiencia previa con estructuras analíticas complejas mejora la percepción de control, fluidez y utilidad de los sistemas basados en IA, lo cual también parece aplicarse en este contexto [149–151].

Adicionalmente, el análisis reveló una asociación estadísticamente significativa entre la experiencia laboral (>7 años) y una mayor valoración de la usabilidad ($p = 0.04462$). Este hallazgo sugiere que no solo el nivel educativo, sino también la trayectoria profesional, influye positivamente en la capacidad de los usuarios para integrarse de forma efectiva con el sistema, fenómeno descrito en la literatura sobre interacción humano-computadora como expertise-based adaptation [152,153].

Complementando la evaluación técnica y de usabilidad, se realizó un análisis semántico-emocional de las percepciones de los usuarios, utilizando redes léxicas generadas a partir de sus expresiones verbales durante el uso del asistente. Este enfoque, aún emergente en la evaluación de sistemas con LLMs, permitió identificar un predominio de emociones positivas como confianza, alegría, anticipación y satisfacción, asociadas semánticamente a términos como «eficaz», «solución», «logro” y «objetivo» [154,155].

Este patrón emocional indica no solo una aceptación racional del sistema, sino también una conexión afectiva favorable, lo cual es clave en entornos de adopción tecnológica. Estudios en neurocognición y HCI (Human-Computer Interaction) han demostrado que la experiencia emocional positiva potencia la adherencia al uso de nuevas tecnologías, mejora la percepción de control y reduce la resistencia al cambio [154,156].

Las emociones negativas (miedo, frustración, confusión) se identificaron en baja frecuencia y estuvieron restringidas a aspectos técnicos puntuales como la manipulación de datos o errores en consola. Esta baja densidad emocional negativa sugiere que el sistema fue, en general, robusto, confiable y cognitivamente amigable [156,157].

Los hallazgos de esta evaluación tienen implicaciones directas para la escalabilidad del sistema y su potencial integración en entornos clínicos y académicos. Primero, la evidencia de usabilidad transversal a distintos niveles de experiencia apoya su utilización en programas de formación, grupos de investigación clínica y redes de colaboración científica. Segundo, la validación emocional refuerza su aceptabilidad, un factor crucial para su adopción voluntaria y sostenida [129,130]. Finalmente, el enfoque HITL utilizado asegura que el sistema no reemplace la capacidad interpretativa del investigador, sino que lo potencia mediante automatización guiada, alineándose con principios éticos y operativos de la IA responsable [158,159].

La implementación de un sistema multiagente basado en Modelos de Lenguaje Grande (LLMs) permitió trascender el enfoque tradicional centrado en automatización, al demostrar empíricamente que es posible integrar a la inteligencia artificial como actor funcional en procesos de investigación científica. A través de la ejecución estructurada de tareas analíticas, el sistema no solo reprodujo flujos de trabajo complejos en R, sino que también participó activamente en la descomposición del problema, la planificación de la solución y la documentación del conocimiento generado. Este tipo de colaboración simbiótica entre humano e IA se alinea con lo propuesto por Shenoi et al. (2023) [160], quienes sostienen que los LLMs tienen potencial epistémico cuando actúan como agentes cognitivos capaces de extender las capacidades inferenciales del investigador, especialmente en contextos de datos abiertos, como los utilizados en este estudio.

Además, los resultados obtenidos respaldan la tesis de que los agentes basados en LLMs pueden integrarse de forma significativa en entornos de cocreación científica, no solo por su eficiencia operativa, sino por su capacidad para estructurar razonamientos, documentar procesos y generar productos replicables y auditables. Estudios recientes como el de Wu et al. (2023) y Chase-Lansdale et al. (2024) [113,161] han destacado que los sistemas de IA que operan bajo un paradigma Human-in-the-Loop no solo automatizan análisis, sino que participan en la arquitectura de la evidencia científica, contribuyendo a la validez metodológica de los hallazgos. El ejercicio aplicado con datos de COVID-19 ejemplifica este enfoque, mostrando cómo una IA puede articular lógica analítica, gestionar incertidumbre y adaptarse a la complejidad de datos reales, lo que consolida su papel no como herramienta pasiva, sino como coautor estructurado en el proceso de construcción del conocimiento.

Capítulo 6: Conclusiones y recomendaciones

6.1 Conclusiones

El presente estudio permitió el desarrollo, validación y evaluación de un sistema multiagente basado en Modelos de Lenguaje Grande (LLMs), concebido para asistir a investigadores biomédicos en la automatización del análisis estadístico. A continuación, se presentan las principales conclusiones, estructuradas en correspondencia con los objetivos planteados:

- Se logró construir una arquitectura modular compuesta por siete agentes especializados —Planificador, Analizador, Investigador, Codificador, Verificador, Depurador y Documentador— capaces de colaborar de forma secuencial e iterativa en tareas de planificación, análisis, generación, verificación y documentación de código. Este diseño permitió una interacción fluida con el usuario en un enfoque human-in-the-loop, que no solo mejoró la precisión del sistema, sino que integró el juicio experto en todo el proceso.

- La validación funcional evidenció un desempeño general robusto. Se obtuvo una alta concordancia interevaluador (AC1 global: 0.831 ± 0.075), y una valoración promedio de 4.38 ± 0.93 puntos sobre 5 en criterios de precisión, replicabilidad y calidad del código. Destacaron ítems como la claridad del código (4.88 ± 0.33) y la documentación técnica (4.88 ± 0.33). No obstante, se identificaron oportunidades de mejora en la identificación automática de variables clave (1.85 ± 0.66) y en la replicación exacta de resultados (3.33 ± 0.69), lo cual sugiere la necesidad de ajustes específicos en estas etapas.

- La prueba del sistema en contextos reales de investigación biomédica reveló una aceptación positiva por parte de los usuarios. El puntaje mediano en la escala de usabilidad SUS fue de 78.8 (IQR: 72.5–87.5), con una clasificación general de

"buena" o "excelente" por el 82% de los participantes. Se observó una relación significativa entre la experiencia laboral y la percepción de usabilidad ($p = 0.044$), indicando que los usuarios más experimentados fueron más proclives a calificar el sistema de forma favorable. Además, el análisis semántico-emocional reflejó un predominio de emociones positivas como confianza, alegría y anticipación, lo cual refuerza la percepción de eficacia y utilidad del asistente.

- El Asistente Multiagente demostró ser especialmente útil para usuarios con competencias limitadas en estadística o programación, al proporcionar herramientas automatizadas y explicaciones interpretables. Esto sugiere un alto potencial para reducir las barreras técnicas que históricamente han limitado la participación de investigadores en etapas avanzadas del análisis de datos, promoviendo así una mayor inclusión académica y equidad en la generación de conocimiento científico.

- La estructura del sistema, basada en la generación de documentación técnica detallada, código replicable y trazabilidad de cada decisión algorítmica, se alinea con los principios de ciencia abierta. Al facilitar la transparencia y la validación cruzada de análisis estadísticos, el Asistente Multiagente se posiciona como una herramienta estratégica para elevar los estándares metodológicos en investigaciones biomédicas, especialmente en contextos donde la reproducibilidad ha sido históricamente deficiente.

- Los resultados obtenidos permiten concluir que el Asistente Multiagente basado en Modelos de Lenguaje Grande cumple con su propósito declarado: asistir de manera estructurada, eficiente y replicable en el análisis estadístico de datos científicos. A través de la aplicación concreta sobre un conjunto de datos reales de COVID-19, se comprobó que el agente no solo ejecuta tareas analíticas automatizadas, sino que también organiza el flujo metodológico, transforma datos complejos y genera productos científicos interpretables. En este sentido, el sistema funciona, evidenciándose cuando fue probado con éxito en un entorno real, confirmando su viabilidad práctica como herramienta de co-creación de conocimiento en investigación biomédica.

6.2 Recomendaciones

Las siguientes recomendaciones surgen como resultado de la experiencia adquirida durante el desarrollo, validación y aplicación del Asistente Multiagente IA basado en LLMs. Están orientadas a fortalecer futuras investigaciones en el área, mejorar las funcionalidades del sistema y ampliar su aplicabilidad en contextos reales de investigación biomédica.

- Se recomienda incorporar técnicas de aprendizaje supervisado con anotación manual de variables clave en artículos científicos, para entrenar modelos que mejoren la capacidad del asistente en reconocer variables dependientes, independientes y de control de forma precisa y automática.

- Es aconsejable extender la funcionalidad del sistema para permitir la integración y análisis de datos multimodales (texto, imágenes médicas, series temporales), lo cual ampliaría su aplicabilidad a estudios de diagnóstico por imagen, genómica o monitoreo fisiológico continuo.

- Se sugiere llevar a cabo pruebas piloto del sistema en contextos hospitalarios o laboratorios clínicos, utilizando datos prospectivos, para evaluar su impacto en tiempo real en la toma de decisiones y generación de reportes biomédicos.

- Es recomendable integrar un mecanismo de aprendizaje continuo (e.g., reinforcement learning o feedback supervisado por el usuario) para que el sistema mejore progresivamente en función de las interacciones humanas y los resultados obtenidos.

- Se aconseja explorar mecanismos de implementación del sistema como plataforma web o API de acceso institucional, lo que permitiría su adopción por grupos de investigación biomédica con diferentes niveles de experiencia técnica.

- Se propone desarrollar una línea de investigación centrada en la inteligencia afectiva, mediante el análisis más profundo de las emociones expresadas por los

usuarios en interacciones prolongadas, con el fin de mejorar la interfaz y aumentar el compromiso del usuario (user engagement).

Referencias Bibliográficas

1 Auger N, Ayoub A, Wei SQ. Letrozole: future alternative to methotrexate for treatment of ectopic pregnancy? *Fertil Steril*. 2020;114:273–4.

2 Gao S, Fang A, Huang Y, *et al.* Empowering biomedical discovery with AI agents. *Cell*. 2024;187:6125–51.

3 Chiarello F, Giordano V, Spada I, *et al.* Future applications of generative large language models: A data-driven case study on ChatGPT. *Technovation*. 2024;133:103002.

4 Brito JJ, Li J, Moore JH, *et al.* Recommendations to enhance rigor and reproducibility in biomedical research. *Gigascience*. 2020;9:1–6.

5 Mische SM, Fisher NC, Meyn SM, *et al.* A review of the scientific rigor, reproducibility, and transparency studies conducted by the ABRF research groups. *J Biomol Tech*. 2020;31:11–26.

6 Ryder J, Leach J. Interpreting experimental data: The views of upper secondary school and university science students. *Int J Sci Educ*. 2000;22:1069–84.

7 Kupczynski M, De Raedt H. Breakdown of statistical inference from some random experiments. *Comput Phys Commun*. 2016;200:168–75.

8 Järvinen TLN, Sihvonen R, Bhandari M, *et al.* Blinded interpretation of study results can feasibly and effectively diminish interpretation bias. *J Clin Epidemiol*. 2014;67:769–72.

9 González-Torres H, Moreno A. Apreciaciones sobre el uso y abuso de la Estadística en Ciencias de la Salud. *Duazary*. 2013;10:62–6.

10 Tiwari K, Kananathan S, Roberts MG, *et al.* Reproducibility in systems biology modelling. *Mol Syst Biol*. 2021;17:1–7.

11 Freedman LP, Cockburn IM, Simcoe TS. The economics of reproducibility in preclinical research. *PLoS Biol*. 2015;13:1–9.

12 Cartes-Velásquez R. The decrease of the scientific budget in Chile for 2018 and the

medical sciences. *Int J Med Surg Sci*. 2018;4:1156–7.

13 Poldrack RA. The Costs of Reproducibility. *Neuron*. 2019;101:11–4.

14 Freedman LP, Inglese J. The increasing urgency for standards in basic biologic research. *Cancer Res*. 2014;74:4024–9.

15 National Academies of Sciences, Engineering, and Medicine; Policy and Global Affairs; Committee on Science, Engineering, Medicine, and Public Policy; Board on Research Data and Information; Division on Engineering and Physical Sciences; Committee on Appli and SSC on R and R in S. *Confidence in Science*. Washington, D.C.: National Academies Press 2019. https://doi.org/10.17226/25303

16 Poldrack RA. The Costs of Reproducibility. *Neuron*. 2019;101:11–4.

17 Deo RC. Machine Learning in Medicine. *Circulation*. 2015;132:1920–30.

18 Rajula HSR, Verlato G, Manchia M, *et al.* Comparison of Conventional Statistical Methods with Machine Learning in Medicine: Diagnosis, Drug Development, and Treatment. *Medicina (Kaunas)*. 2020;56. doi: 10.3390/medicina56090455

19 Martí-Juan G, Sanroma-Guell G, Piella G. A survey on machine and statistical learning for longitudinal analysis of neuroimaging data in Alzheimer's disease. *Comput Methods Programs Biomed*. 2020;189:105348.

20 Archibald MM, Clark AM. ChatGTP: What is it and how can nursing and health science education use it? *J Adv Nurs*. 2023;79:3648–51.

21 Cohen A, Alter R, Lessans N, *et al.* Performance of ChatGPT in Israeli Hebrew OBGYN national residency examinations. *Arch Gynecol Obstet*. 2023;308:1797–802.

22 King RC, Samaan JS, Yeo YH, *et al.* A Multidisciplinary Assessment of ChatGPT's Knowledge of Amyloidosis: Observational Study. *JMIR cardio*. 2024;8:e53421.

23 Xu X, Chen Y, Miao J. Opportunities, challenges, and future directions of large language models, including ChatGPT in medical education: a systematic scoping review. *J Educ Eval Health Prof*. 2024;21:6.

24 Sirocchi C, Bogliolo A, Montagna S. Medical-informed machine learning: integrating prior knowledge into medical decision systems. *BMC Med Inform Decis Mak*. 2024;24:186.

25 Singhal K, Azizi S, Tu T, *et al.* Large language models encode clinical knowledge. *Nature*. 2023;620:172–80.

26 Alonso I, Oronoz M, Agerri R. MedExpQA: Multilingual benchmarking of Large

Language Models for Medical Question Answering. *Artif Intell Med*. 2024;155:102938.

27 Liévin V, Hother CE, Motzfeldt AG, *et al.* Can large language models reason about medical questions? *Patterns (New York, NY)*. 2024;5:100943.

28 Contreras Kallens P, Kristensen-McLachlan RD, Christiansen MH. Large Language Models Demonstrate the Potential of Statistical Learning in Language. *Cogn Sci*. 2023;47:e13256.

29 Dirnagl U, Duda GN, Grainger DW, *et al.* Reproducibility, relevance and reliability as barriers to efficient and credible biomedical technology translation. *Adv Drug Deliv Rev*. 2022;182:114118.

30 Weissler EH, Naumann T, Andersson T, *et al.* The role of machine learning in clinical research: transforming the future of evidence generation. *Trials*. 2021;22:537.

31 Noorbakhsh-Sabet N, Zand R, Zhang Y, *et al.* Artificial Intelligence Transforms the Future of Health Care. *Am J Med*. 2019;132:795–801.

32 Hulsen T, Manni F. Guest Editorial: Big data and artificial intelligence in healthcare. *Healthc Technol Lett*. 2024;11:207–9.

33 Kamath U, Keenan K, Somers G, *et al.* Large Language Models: An Introduction. *Large Language Models: A Deep Dive*. Cham: Springer Nature Switzerland 2024:1–27. https://doi.org/10.1007/978-3-031-65647-7_1

34 Carpenter CR, Griffey RT, Rutjes AWS, *et al.* Problem with the existing reporting standards for adverse event and medical error research. *BMJ Qual Saf*. Published Online First: February 2025. doi: 10.1136/bmjqs-2024-017491

35 Wong IN, Monteiro O, Baptista-Hon DT, *et al.* Leveraging foundation and large language models in medical artificial intelligence. *Chin Med J (Engl)*. 2024;137:2529–39.

36 Gao C, Lan X, Li N, *et al.* Large language models empowered agent-based modeling and simulation: a survey and perspectives. *Humanit Soc Sci Commun*. 2024;11:1259.

37 Nazi Z Al, Peng W. Large Language Models in Healthcare and Medical Domain: A Review. *Informatics*. 2024;11:57.

38 Jiang LY, Liu XC, Nejatian NP, *et al.* Health system-scale language models are all-purpose prediction engines. *Nature*. 2023;619:357–62.

39 Ferrara E. Large Language Models for Wearable Sensor-Based Human Activity Recognition, Health Monitoring, and Behavioral Modeling: A Survey of Early Trends, Datasets, and Challenges. *Sensors (Basel)*. 2024;24. doi: 10.3390/s24155045

40 Jin Q, Leaman R, Lu Z. PubMed and beyond: biomedical literature search in the age of artificial intelligence. *eBioMedicine*. 2024;100:104988.

41 Lee J, Yoon W, Kim SS, *et al.* BioBERT: a pre-trained biomedical language representation model for biomedical text miningtle. *Bioinformatics*. 2020;36:1234–40.

42 Myszewski JJ, Klossowski E, Meyer P, *et al.* Validating GAN-BioBERT: A Methodology for Assessing Reporting Trends in Clinical Trials. *Front Digit Heal*. 2022;4:878369.

43 Manchev N. Generative AI: A Case Study Highlighting the Potential and the Risks Translating legacy SAS code to modern open-source alternatives. *Domino Data Lab*. 2023;1–23. https://spectrum.ieee.org/top-programming-languages-methodology

44 Khan AE, Hasan MJ, Anjum H, *et al.* Predicting life satisfaction using machine learning and explainable AI. *Heliyon*. 2024;10:e31158.

45 Shinde PP, Shah S. A Review of Machine Learning and Deep Learning Applications. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. IEEE 2018:1–6. https://doi.org/10.1109/ICCUBEA.2018.8697857

46 Harrington P. *Machine Learning in Action*. Shelter Island, NY: Manning Publications Co 2012.

47 Bi Q, Goodman KE, Kaminsky J, *et al.* What is Machine Learning? A Primer for the Epidemiologist. *Am J Epidemiol*. 2019;188:2222–39.

48 Roth JA, Battegay M, Juchler F, *et al.* Introduction to Machine Learning in Digital Healthcare Epidemiology. *Infect Control Hosp Epidemiol*. 2018;39:1457–62.

49 S K S, P A. A Machine Learning Ensemble Classifier for Early Prediction of Diabetic Retinopathy. *J Med Syst*. 2017;41:201.

50 Serra-Burriel M, Ames C. Machine Learning-Based Clustering Analysis: Foundational Concepts, Methods, and Applications. *Acta Neurochir Suppl*. 2022;134:91–100.

51 Hou J, Gao H, Li X. DSets-DBSCAN: A Parameter-Free Clustering Algorithm. *IEEE*

Trans Image Process. 2016;25:3182–93.

52 Yamasaki R, Tanaka T. Properties of Mean Shift. *IEEE Trans Pattern Anal Mach Intell*. 2020;42:2273–86.

53 Timmerman ME, Ceulemans E, De Roover K, *et al.* Subspace K-means clustering. *Behav Res Methods*. 2013;45:1011–23.

54 Steinley D. K-means clustering: a half-century synthesis. *Br J Math Stat Psychol*. 2006;59:1–34.

55 Kusy M. Fuzzy c-means-based architecture reduction of a probabilistic neural network. *Neural Netw*. 2018;108:20–32.

56 Naeem S, Ali A, Anam S, *et al.* An Unsupervised Machine Learning Algorithms: Comprehensive Review. *Int J Comput Digit Syst*. 2023;13:911–21.

57 Khanum M, Mahboob T, Imtiaz W, *et al.* A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance. *Int J Comput Appl*. 2015;119:34–9.

58 Tomašev N, Radovanović M. Clustering Evaluation in High-Dimensional Data. *Unsupervised Learning Algorithms*. Cham: Springer International Publishing 2016:71–107. https://doi.org/10.1007/978-3-319-24211-8_4

59 Eckhardt CM, Madjarova SJ, Williams RJ, *et al.* Unsupervised machine learning methods and emerging applications in healthcare. *Knee Surg Sports Traumatol Arthrosc*. 2023;31:376–81.

60 Martin DI, Martin JC, Berry MW. The Application of LSA to the Evaluation of Questionnaire Responses. *Unsupervised Learning Algorithms*. Cham: Springer International Publishing 2016:449–84. https://doi.org/10.1007/978-3-319-24211-8_16

61 Kalcheva N, Todorova M, Marinova G. Naive Bayes Classifier, Decision Tree And Adaboost Ensemble Algorithm – Advantages And Disadvantage. 2020:153–7. https://doi.org/10.31410/ERAZ.2020.153

62 Jiang T, Gradus JL, Rosellini AJ. Supervised Machine Learning: A Brief Primer. *Behav Ther*. 2020;51:675–87.

63 Bechelli S, Delhommelle J. Machine Learning and Deep Learning Algorithms for Skin Cancer Classification from Dermoscopic Images. *Bioeng (Basel, Switzerland)*. 2022;9. doi: 10.3390/bioengineering9030097

64 Uddin S, Khan A, Hossain ME, *et al.* Comparing different supervised machine

learning algorithms for disease prediction. *BMC Med Inform Decis Mak*. 2019;19:281.

65 Lo Vercio L, Amador K, Bannister JJ, *et al.* Supervised machine learning tools: a tutorial for clinicians. *J Neural Eng*. 2020;17. doi: 10.1088/1741-2552/abbff2

66 Choudhary R, Gianey HK. Comprehensive Review On Supervised Machine Learning Algorithms. *2017 International Conference on Machine Learning and Data Science (MLDS)*. IEEE 2017:37–43. https://doi.org/10.1109/MLDS.2017.11

67 Serghiou S, Rough K. Deep Learning for Epidemiologists: An Introduction to Neural Networks. *Am J Epidemiol*. 2023;192:1904–16.

68 Gabrié M, Ganguli S, Lucibello C, *et al.* Neural Networks: From the Perceptron to Deep Nets. *Spin Glass Theory and Far Beyond*. WORLD SCIENTIFIC 2023:477–97. https://doi.org/10.1142/9789811273926_0024

69 Gallant SI. Perceptron-based learning algorithms. *IEEE Trans Neural Networks*. 1990;1:179–91.

70 Chen S, Guo W. Auto-Encoders in Deep Learning—A Review with New Perspectives. *Mathematics*. 2023;11:1777.

71 Berahmand K, Daneshfar F, Salehi ES, *et al.* Autoencoders and their applications in machine learning: a survey. *Artif Intell Rev*. 2024;57:28.

72 Palasundram K, Mohd Sharef N, Kasmiran KA, *et al.* Enhancements to the Sequence-to-Sequence-Based Natural Answer Generation Models. *IEEE Access*. 2020;8:45738–52.

73 Wenzel M. *Generative Adversarial Networks and Other Generative Models*. 2023. http://www.ncbi.nlm.nih.gov/pubmed/18688245

74 Yoon JT, Lee KM, Oh J-H, *et al.* Insights and Considerations in Development and Performance Evaluation of Generative Adversarial Networks (GANs): What Radiologists Need to Know. *Diagnostics (Basel, Switzerland)*. 2024;14. doi: 10.3390/diagnostics14161756

75 Jeong JJ, Tariq A, Adejumo T, *et al.* Systematic Review of Generative Adversarial Networks (GANs) for Medical Image Classification and Segmentation. *J Digit Imaging*. 2022;35:137–52.

76 Cossu A, Carta A, Lomonaco V, *et al.* Continual learning for recurrent neural networks: An empirical evaluation. *Neural Netw*. 2021;143:607–27.

77 Kriegeskorte N, Golan T. Neural network models and deep learning. *Curr Biol*.

2019;29:R231–6.

78 Student MT, Kumar RR, Omments REC, *et al.* Lupus Nephritis. En Comprehensive Clinical. *Front Neurosci*. 2021;14:1–13.

79 Dockendorf KP, Park I, He P, *et al.* Liquid state machines and cultured cortical networks: the separation property. *Biosystems*. 2009;95:90–7.

80 Van Houdt G, Mosquera C, Nápoles G. A review on the long short-term memory model. *Artif Intell Rev*. 2020;53:5929–55.

81 Hochreiter S, Schmidhuber J. Long Short-Term Memory. *Neural Comput*. 1997;9:1735–80.

82 Oğuz A, Ertuğrul ÖF. Introduction to deep learning and diagnosis in medicine. *Diagnostic Biomedical Signal and Image Processing Applications with Deep Learning Methods*. Elsevier 2023:1–40. https://doi.org/10.1016/B978-0-323-96129-5.00003-2

83 Deepak GD, Bhat SK, Gupta A. Improved CNN architecture for automated classification of skin diseases. *Comput Methods Biomech Biomed Eng Imaging Vis*. 2025;13. doi: 10.1080/21681163.2024.2420727

84 Aloysius N, Geetha M. A review on deep convolutional neural networks. *2017 International Conference on Communication and Signal Processing (ICCSP)*. IEEE 2017:0588–92. https://doi.org/10.1109/ICCSP.2017.8286426

85 Belyadi H, Haghighat A. Machine learning workflows and types. *Machine Learning Guide for Oil and Gas Using Python*. Elsevier 2021:97–123. https://doi.org/10.1016/B978-0-12-821929-4.00001-9

86 Jia J, Wang W. Review of reinforcement learning research. *2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE 2020:186–91. https://doi.org/10.1109/YAC51587.2020.9337653

87 Belyadi H, Haghighat A. Machine learning workflows and types. *Machine Learning Guide for Oil and Gas Using Python*. Elsevier 2021:97–123. https://doi.org/10.1016/B978-0-12-821929-4.00001-9

88 Shobha G, Rangaswamy S. Machine Learning. 2018:197–228. https://doi.org/10.1016/bs.host.2018.07.004

89 Jia J, Wang W. Review of reinforcement learning research. *2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE 2020:186–91. https://doi.org/10.1109/YAC51587.2020.9337653

90 Katoch S, Chauhan SS, Kumar V. A review on genetic algorithm: past, present, and future. *Multimed Tools Appl*. 2021;80:8091–126.

91 Watkins CJCH, Dayan P. Q-learning. *Mach Learn*. 1992;8:279–92.

92 Lamba D, Hsu WH, Alsadhan M. Predictive analytics and machine learning for medical informatics: A survey of tasks and techniques. *Machine Learning, Big Data, and IoT for Medical Informatics*. Elsevier 2021:1–35. https://doi.org/10.1016/B978-0-12-821777-1.00023-9

93 Watkins CJCH, Dayan P. Q-learning. *Mach Learn*. 1992;8:279–92.

94 Moghaddasi K, Rajabi S, Gharehchopogh FS. An enhanced asynchronous advantage actor-critic-based algorithm for performance optimization in mobile edge computing -enabled internet of vehicles networks. *Peer-to-Peer Netw Appl*. 2024;17:1169–89.

95 Thirunavukarasu AJ, Ting DSJ, Elangovan K, *et al.* Large language models in medicine. *Nat Med*. 2023;29:1930–40.

96 Dakhel AM, Nikanjam A, Khomh F, *et al.* An Overview on Large Language Models. *Generative AI for Effective Software Development*. Cham: Springer Nature Switzerland 2024:3–21. https://doi.org/10.1007/978-3-031-55642-5_1

97 Lobentanzer S, Feng S, Bruderer N, *et al.* A platform for the biomedical application of large language models. *Nat Biotechnol*. Published Online First: January 2025. doi: 10.1038/s41587-024-02534-3

98 Vaswani A, Shazeer N, Parmar N, *et al.* Attention is all you need. *Adv Neural Inf Process Syst*. 2017;2017-Decem:5999–6009.

99 Anisuzzaman DM, Malins JG, Friedman PA, *et al.* Fine-Tuning Large Language Models for Specialized Use Cases. *Mayo Clin Proc Digit Heal*. 2025;3:100184.

100 Goswami J, Prajapati KK, Saha A, *et al.* Parameter-efficient fine-tuning large language model approach for hospital discharge paper summarization. *Appl Soft Comput*. 2024;157:111531.

101 Peng C, Yang X, Smith KE, *et al.* Model tuning or prompt Tuning? a study of large language models for clinical concept and relation extraction. *J Biomed Inform*. 2024;153:104630.

102 Hu Y, Chen Q, Du J, *et al.* Improving large language models for clinical named entity recognition via prompt engineering. *J Am Med Inform Assoc*. 2024;31:1812–20.

103 Svensson A. *What is the best API from a developer's perspective? Investigation of API development with fintech developers in the spotlight*. 2024. https://www.diva-portal.org/smash/get/diva2:1865779/FULLTEXT02se

104 Kim M, Stennett T, Shah D, *et al.* Leveraging Large Language Models to Improve REST API Testing. *Proc - Int Conf Softw Eng*. 2024;37–41.

105 Brunette ES, Flemmer RC, Flemmer CL. A review of artificial intelligence. *2009 4th International Conference on Autonomous Robots and Agents*. IEEE 2009:385–92. https://doi.org/10.1109/ICARA.2000.4804025

106 Huang KA, Choudhary HK, Kuo PC. Artificial Intelligent Agent Architecture and Clinical Decision-Making in the Healthcare Sector. *Cureus*. 2024;16:e64115.

107 Brunette ES, Flemmer RC, Flemmer CL. A review of artificial intelligence. *2009 4th International Conference on Autonomous Robots and Agents*. IEEE 2009:385–92. https://doi.org/10.1109/ICARA.2000.4804025

108 Hendler JA. Intelligent Agents: Where AI Meets Information Technology [Guest Editorial]. *IEEE Expert*. 1996;11:20.

109 Johnson J, Douze M, Jegou H. Billion-Scale Similarity Search with GPUs. *IEEE Trans Big Data*. 2021;7:535–47.

110 Lewis P, Perez E, Piktus A, *et al.* Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Published Online First: 22 May 2020.

111 Izacard G, Grave E. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Stroudsburg, PA, USA: Association for Computational Linguistics 2021:874–80. https://doi.org/10.18653/v1/2021.eacl-main.74

112 Barr AA, Quan J, Guo E, *et al.* Large language models generating synthetic clinical datasets: a feasibility and comparative analysis with real-world perioperative data. *Front Artif Intell*. 2025;8:1533508.

113 Wu J, Huang Z, Hu Z, *et al.* Toward Human-in-the-Loop AI: Enhancing Deep Reinforcement Learning via Real-Time Human Guidance for Autonomous Driving. *Engineering*. 2023;21:75–91.

114 Wang D, Zhang S. Large language models in medical and healthcare fields: applications, advances, and challenges. *Artif Intell Rev*. 2024;57:299.

115 Mondal H, Mondal S, Mittal P. Evaluating large language models for selection of

statistical test for research: A pilot study. *Perspect Clin Res*. 2024;15:178–82.

116 Busch F, Hoffmann L, Rueger C, *et al.* Current applications and challenges in large language models for patient care: a systematic review. *Commun Med*. 2025;5:26.

117 Rakauskas TR, Da Costa A, Moriconi C, *et al.* Evaluation of Chat Generative Pre-trained Transformer and Microsoft Copilot Performance on the American Society of Surgery of the Hand Self-Assessment Examinations. *J hand Surg Glob online*. 2025;7:23–8.

118 Ding L, Fan L, Shen M, *et al.* Evaluating ChatGPT's diagnostic potential for pathology images. *Front Med*. 2024;11:1507203.

119 Hirosawa T, Harada Y, Tokumasu K, *et al.* Evaluating ChatGPT-4's Diagnostic Accuracy: Impact of Visual Data Integration. *JMIR Med informatics*. 2024;12:e55627.

120 Wang X, Ye H, Zhang S, *et al.* Evaluation of the Performance of Three Large Language Models in Clinical Decision Support: A Comparative Study Based on Actual Cases. *J Med Syst*. 2025;49:23.

121 Chen W, Su Y, Zuo J, *et al.* AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. Published Online First: 21 August 2023. doi: 10.48550/arXiv.2308.10848

122 Hong S, Zhuge M, Chen J, *et al.* MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. Published Online First: 1 August 2023.

123 Servantez S, Barrow J, Hammond K, *et al.* Chain of Logic: Rule-Based Reasoning with Large Language Models. *Findings of the Association for Computational Linguistics ACL 2024*. Stroudsburg, PA, USA: Association for Computational Linguistics 2024:2721–33. https://doi.org/10.18653/v1/2024.findings-acl.159

124 Liu X, Lou X, Jiao J, *et al.* Position: Foundation Agents as the Paradigm Shift for Decision Making. Published Online First: 27 May 2024.

125 Kojima T, Gu SS, Reid M, *et al.* Large Language Models are Zero-Shot Reasoners. Published Online First: 24 May 2022.

126 Amershi S, Cakmak M, Knox WB, *et al.* Power to the People: The Role of Humans in Interactive Machine Learning. *AI Mag*. 2014;35:105–20.

127 Holzinger A. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain informatics*. 2016;3:119–31.

128 Raji ID, Smart A, White RN, *et al.* Closing the AI accountability gap. *Proceedings of*

the 2020 Conference on Fairness, Accountability, and Transparency. New York, NY, USA: ACM 2020:33–44. https://doi.org/10.1145/3351095.3372873

129 Topol EJ. High-performance medicine: the convergence of human and artificial intelligence. *Nat Med*. 2019;25:44–56.

130 Goodman KW. Ethics in Health Informatics. *Yearb Med Inform*. 2020;29:26–31.

131 Gwet KL. *Handbook of Inter-Rater Reliability Fourth Edition*. 4th ed. Gaithersburg: Advanced Analytics, LLC 2014.

132 Salimiparsa M, Lizotte D, Sedig K. A User-Centered Design of Explainable AI for Clinical Decision Support. *Proc Can Conf Artif Intell*. Published Online First: 8 June 2021. doi: 10.21428/594757db.62860442

133 Kiyasseh D, Cohen A, Jiang C, *et al.* A framework for evaluating clinical artificial intelligence systems without ground-truth annotations. *Nat Commun*. 2024;15:1808.

134 Huynh N, Lin B. Large Language Models for Code Generation: A Comprehensive Survey of Challenges, Techniques, Evaluation, and Applications. Published Online First: 3 March 2025. doi: 10.48550/arXiv.2503.01245

135 Du X, Liu M, Wang K, *et al.* Evaluating Large Language Models in Class-Level Code Generation. *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. New York, NY, USA: ACM 2024:1–13. https://doi.org/10.1145/3597503.3639219

136 Munafò MR, Nosek BA, Bishop DVM, *et al.* A manifesto for reproducible science. *Nat Hum Behav*. 2017;1:0021.

137 Peng RD. Reproducible research in computational science. *Science*. 2011;334:1226–7.

138 Goodman SN, Fanelli D, Ioannidis JPA. What does research reproducibility mean? *Sci Transl Med*. 2016;8:341ps12.

139 Liu P, Yuan W, Fu J, *et al.* Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput Surv*. 2023;55. doi: 10.1145/3560815

140 Košprdić M, Prodanović N, Ljajić A, *et al.* From zero to hero: Harnessing transformers for biomedical named entity recognition in zero- and few-shot contexts. *Artif Intell Med*. 2024;156:102970.

141 Bodenreider O. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Res*. 2004;32:D267-70.

142 Shneiderman B. Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy. *Int J Human–Computer Interact*. 2020;36:495–504.

143 Chen X, Xiang J, Lu S, *et al.* Evaluating large language models and agents in healthcare: key challenges in clinical applications. *Intell Med*. 2025;5:151–63.

144 Saleh Y, Abu Talib M, Nasir Q, *et al.* Evaluating large language models: a systematic review of efficiency, applications, and future directions. *Front Comput Sci*. 2025;7. doi: 10.3389/fcomp.2025.1523699

145 Fan A, Gokkaya B, Harman M, *et al.* Large Language Models for Software Engineering: Survey and Open Problems. *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. IEEE 2023:31–53. https://doi.org/10.1109/ICSE-FoSE59343.2023.00008

146 Elliott JH, Turner T, Clavisi O, *et al.* Living systematic reviews: an emerging opportunity to narrow the evidence-practice gap. *PLoS Med*. 2014;11:e1001603.

147 Sudha S, Sunil SK, Parthiv Akilesh AS, *et al.* Democratizing Data Science:Using Language Models for Intuitive Data Insights and Visualizations. *2024 4th International Conference on Pervasive Computing and Social Networking (ICPCSN)*. IEEE 2024:1065–9. https://doi.org/10.1109/ICPCSN62568.2024.00177

148 Köpf A, Kilcher Y, von Rütte D, *et al.* OpenAssistant Conversations -- Democratizing Large Language Model Alignment. *37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks.* 2023:1–13. http://arxiv.org/abs/2304.07327

149 Borsci S, Federici S, Lauriola M. On the dimensionality of the System Usability Scale: a test of alternative measurement models. *Cogn Process*. 2009;10:193–7.

150 Bangor A, Kortum PT, Miller JT. An Empirical Evaluation of the System Usability Scale. *Int J Hum Comput Interact*. 2008;24:574–94.

151 Wekenborg MK, Gilbert S, Kather JN. Examining human-AI interaction in real-world healthcare beyond the laboratory. *npj Digit Med*. 2025;8:169.

152 Norman D. *The Design of Everyday Things*. 2013. https://doi.org/10.1145/1340961.1340979

153 Winkler R, Söllner M, Leimeister JM. Enhancing problem-solving skills with smart personal assistant technology. *Comput Educ*. 2021;165:104148.

154 Calvo RA, D'Mello S. Affect Detection: An Interdisciplinary Review of Models, Methods, and Their Applications. *IEEE Trans Affect Comput*. 2010;1:18–37.

155 Mohammad SM, Turney PD. Crowdsourcing a word–emotion association lexicon. *Comput Intell*. 2013;29:436–65.

156 Peter C, Urban B. Emotion in Human-Computer Interaction. *Expanding the Frontiers of Visual Analytics and Visualization*. London: Springer London 2012:239–62. https://doi.org/10.1007/978-1-4471-2804-5_14

157 Afzal S, Ali Khan H, Jalil Piran M, *et al.* A Comprehensive Survey on Affective Computing: Challenges, Trends, Applications, and Future Directions. *IEEE Access*. 2024;12:96150–68.

158 European Medicines Agency. Reflection paper on the use of Artificial Intelligence (AI) in the medicinal product lifecycle. 2023. https://www.ema.europa.eu/en/documents/scientific-guideline/draft-reflection-paper-use-artificial-intelligence-ai-medicinal-product-lifecycle_en.pdf

159 Kenthapadi K, Sameki M, Taly A. Grounding and Evaluation for Large Language Models: Practical Challenges and Lessons Learned (Survey). *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM 2024:6523–33. https://doi.org/10.1145/3637528.3671467

160 Shenoi VV, Sreeram P, Sai Varma CL, *et al.* Analysing the Role of Human-AI Collaboration in Workforce Transformation. *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*. IEEE 2024:1–7. https://doi.org/10.1109/ADICS58448.2024.10533640

161 Falcone R, Castelfranchi C. The human in the loop of a delegated agent: the theory of adjustable social autonomy. *IEEE Trans Syst Man, Cybern - Part A Syst Humans*. 2001;31:406–18.

A. Anexo: Instrumento de Evaluación Heurística–Funcional

Nº	Q	Descriptor	Criterio de evaluación funcional	Descripción	Puntaje (1–5)	Observaciones
1	Q_01	Correspondencia	Correspondencia del código generado con los métodos del artículo	Evalúa si el código generado refleja adecuadamente las técnicas estadísticas descritas en la sección de métodos del paper		
2	Q_02	Replicacion	Replicación de resultados (tablas, figuras, coeficientes)	Verifica si el código reproduce fielmente los resultados del artículo (números, modelos, gráficos)		
3	Q_03	Identificacion variables	Identificación automática de variables clave	Evalúa si el asistente identifica correctamente las variables dependientes, independientes y de agrupación		
4	Q_04	Modelo estadistico	Elección adecuada del modelo estadístico	Evalúa si el asistente selecciona el tipo de modelo correcto (lineal, logístico, Cox, GLM, etc.) según el diseño del estudio		
5	Q_05	Preprocesamiento	Inclusión de pasos de preprocesamiento	Evalúa si se integran adecuadamente transformaciones como imputación, normalización o exclusión de casos		
6	Q_06	Claridad codigo	Claridad del código generado	Verifica si el código es legible, estructurado, modular y con comentarios explicativos		
7	Q_07	Documentacion	Documentación técnica generada	Evalúa si se incluye documentación adicional con explicaciones, fórmulas en LaTeX y razonamiento estadístico		

Nº	Q	Descriptor	Criterio de evaluación funcional	Descripción	Puntaje (1–5)	Observaciones
8	Q_08	Tiempo validacion	Tiempo de evaluación/validación reducido	Evalúa el grado en que el asistente facilita y acelera la verificación o reproducción del análisis del artículo		
9	Q_09	Ambiguedades	Manejo de ambigüedades	Determina si el asistente alerta o resuelve ambigüedades en nombres de variables o decisiones analíticas no descritas		
10	Q_10	Utilidad asistente	Utilidad percibida del asistente para automatizar la validación científica	Valoración general del evaluador sobre el valor práctico del sistema para automatizar análisis y validaciones de papers		
11	Q_11	Ejecucion fuera Streamlit	El código generado puede ejecutarse y validarse fácilmente fuera del entorno Streamlit	Evalúa la portabilidad y compatibilidad del código con otros entornos como RStudio o Jupyter		
12	Q_12	Sin trazas consola	El sistema no muestra trazas innecesarias por consola	Evalúa si el sistema evita mensajes de depuración innecesarios que dificulten su uso		
13	Q_13	Refinamiento post ejecucion	El sistema permite continuar con refinamiento del código o documentación tras la ejecución	Evalúa si el flujo multiagente permite una secuencia coherente desde código hasta documentación final		
14	Q_14	Visualizacion interfaz	Visualización del código generado en la interfaz	Evalúa si el código se muestra visualmente en Streamlit con opción de copia y revisión		
15	Q_15	Preguntas codigo doc	Preguntas sobre el código y documentación	Evalúa si el sistema permite formular preguntas y ofrece respuestas precisas y útiles		

B. Anexo: Evaluación de Usabilidad del Agente de Análisis Estadístico con LLMs

Evaluación de Usabilidad del Agente de Análisis Estadístico con LLMs

Este formulario tiene como propósito recopilar la percepción de usabilidad del agente inteligente desarrollado en el marco del proyecto *"Implementación del Análisis Estadístico en Ciencias Biomédicas mediante Agentes Basados en Modelos de Lenguaje Grande (LLMs)"*.

A través de la escala estandarizada **System Usability Scale (SUS)**, se evaluarán aspectos clave como la facilidad de uso, eficiencia operativa y satisfacción general del usuario. Su participación es fundamental para validar la aplicabilidad práctica del sistema en entornos reales de investigación biomédica.

Las respuestas serán tratadas de forma confidencial y empleadas exclusivamente con fines de mejora continua y validación académica del sistema.

Acepto participar en la evaluación de usabilidad del agente inteligente basado en LLMs y autorizo el uso de mis respuestas para fines académicos y científicos.

☐ Acepto ☐ No Acepto

Información Sociodemográfica: _____
Identificación: _____
Edad (años cumplidos): _____

Sexo

Femenino: ☐ Masculino: ☐

Nivel educativo alcanzado

Pregrado: ☐ Especialización: ☐ Maestría: ☐ Doctorado: ☐

Área de formación profesional o disciplinar

Ciencias biomédicas ☐

Salud pública / Epidemiología ☐

Estadística / Bioestadística / Ciencia de Datos ☐

Ciencias computacionales ☐

Años de experiencia en investigación científica:

Ninguno: ☐ 1–3 años: ☐ 4–7 años: ☐ Más de 7 años: ☐

¿Con qué frecuencia realiza análisis estadísticos en su trabajo/investigación?

Poco: ☐ Ocasionalmente: ☐ Frecuentemente: ☐ Siempre: ☐

Conocimiento previo en herramientas de análisis estadístico

	Ninguno	Básico	Intermedio	Avanzado
SPSS	☐	☐	☐	☐
STATA	☐	☐	☐	☐
R	☐	☐	☐	☐
Python	☐	☐	☐	☐

¿Había utilizado previamente herramientas basadas en inteligencia artificial para análisis estadístico?

Sí ☐ No ☐

¿Había trabajado previamente con Modelos de Lenguaje Grande (LLMs) tales como ChatGTP, Gemini, Perplexity, DeepSeek?

Sí ☐ No ☐

System Usability Scale (SUS)

Totalmente en desacuerdo (1); En desacuerdo (2); Neutral (3); De acuerdo (4); Totalmente de acuerdo (5)

1. Creo que me gustaría utilizar este agente inteligente para análisis estadístico con frecuencia en mis investigaciones biomédicas.
2. Me pareció que el agente inteligente era extremadamente complejo para los análisis que realizo.
3. Considero que el agente fue fácil de utilizar para ejecutar análisis estadísticos.

4. Pienso que necesitaría el apoyo de una persona experta en informática o estadística para poder utilizar correctamente este agente.
5. Observé que las diferentes funciones del agente estaban bien integradas y coordinadas.
6. Me pareció que había demasiada inconsistencia en el funcionamiento del agente.
7. Me imagino que la mayoría de los investigadores biomédicos aprenderían a utilizar este agente con rapidez.
8. Considero que el agente fue difícil de usar para realizar los análisis requeridos.
9. Me sentí seguro al utilizar el agente para realizar análisis estadísticos.
10. Sentí que necesitaba aprender muchas cosas antes de poder comenzar a usar correctamente el agente para el análisis estadístico.

Evaluación Cualitativa

A continuación, encontrará tres preguntas abiertas diseñadas para conocer su experiencia en términos de **eficacia, eficiencia y satisfacción**. Le solicitamos que responda **en un mensaje de voz lo más detallado posible** (2 a 5 minutos) identificándose con tu número, en el que comparta sus percepciones personales sobre el uso del sistema.

Por favor, grabe su respuesta y envíela vía WhatsApp al número: +57 301 565 7574

Eficacia: Explique si logró los resultados esperados y cómo fue ese proceso. ¿Cómo describiría su experiencia al intentar alcanzar sus objetivos de análisis estadístico utilizando el agente inteligente?

Eficiencia: Describa cualquier dificultad que haya encontrado o aspectos que facilitaron su uso. ¿Qué tan demandante le resultó utilizar el agente en términos de tiempo, esfuerzo o necesidad de apoyo adicional?

Satisfacción: Comparta su percepción general y sugerencias de mejora si las tiene. ¿Qué aspectos del uso del agente le resultaron más agradables o frustrantes durante su experiencia?

C. Anexo: Continuación del Cuadro 13.– Representación esquemática del plan de investigación por el Agente Investigador

- Punto B: Creación de variables de intervalo de tiempo
  - dias_diagnostico:Calcular como la diferencia en días entre fecha_diagnosticoy fecha_inicio_sintomas.
  - dias_recuperacion:Calcula como la diferencia entre fecha_recuperadoy fecha_diagnostico.
  - dias_muerte:Calcula como la diferencia entre fecha_muertey fecha_diagnostico.
  - Utilice mutate()desde dplyr y asegúrese de manejar valores NA (las diferencias deben ser NA si faltan una o ambas fechas).
- Punto C: Análisis descriptivo de la edad por sexo
  - Utilice group_by(sexo)y summarise()para calcular la media, mediana, mínimo y máximo edadpor sexo.
  - Grafique histogramas y diagramas de caja edadutilizando sexoggplot2: geom_histogram()y geom_boxplot(), utilizando fillo colorestética para separar por sexo.
  - Compruebe si hay valores atípicos o inconsistencias en la edad (por ejemplo, edades improbables).
- Punto D: Tablas de frecuencias por sexo
  - Para departamento_nom, per_etn_, y nom_grupo_, cree tablas de frecuencia utilizando table()o dplyr count()con group_by(sexo, variable).
  - Mostrar frecuencias absolutas y relativas (utilizar prop.table()o calcular proporciones manualmente).
  - Esté atento a nombres de categorías faltantes o inconsistentes (por ejemplo, errores tipográficos, ortografías diferentes).
- Punto E: Análisis del tiempo clínico por sexo
  - Para cada una de las nuevas variables (dias_diagnostico, dias_recuperacion, dias_muerte), calcule la media, la mediana, el mínimo y el máximo por sexo.
  - Visualice distribuciones utilizando ggplot2: diagramas de caja y diagramas de densidad, separados por sexo.
  - Investigue la presencia de intervalos negativos o inverosímiles, que pueden indicar problemas de calidad de los datos.
- Punto F: Tabulaciones cruzadas por sexo
  - Para estado, ubicacion, tipo_recuperacion, y recuperado, cree tablas de contingencia cruzando cada variable con sexo.
  - Muestra tanto los recuentos como las proporciones (porcentajes de fila/columna según corresponda).
  - Utilice table(), prop.table(), y/o janitor::tabyl()para tablas limpias.
- Punto G: Gráficos de evolución semanal por sexo

    - Agrupe los datos por semana (floor_date(fecha_diagnostico, "week"), etc.) y sexo, luego cuente los casos, las recuperaciones y las muertes por semana.
    - Grafique series de tiempo con ggplot2, usando geom_line()o geom_col()para cada tipo de evento, coloreado por sexo.
    - Asegúrese de que todas las variables de fecha se manejen adecuadamente para los valores faltantes.
- Punto H: Visualización en RStudio
    - Todas las tablas se pueden imprimir print()o visualizar de forma interactiva en RStudio.
    - Todos los gráficos generados con ggplot2 se pueden mostrar en el panel Gráficos de RStudio.

Posibles desafíos:

- Los valores de fecha faltantes o inconsistentes pueden generar NA o intervalos negativos: estos deben verificarse y posiblemente filtrarse.
- Las variables categóricas (por ejemplo, sexo, departamento, etnia) pueden tener inconsistencias o etiquetas faltantes.
- En algunos casos puede faltar información sobre resultados secundarios (por ejemplo, fechas de recuperación o de muerte), lo que afecta los análisis de intervalos de tiempo.
- Los conjuntos de datos grandes pueden requerir funciones que hagan un uso eficiente de la memoria y un uso cuidadoso de las canalizaciones dplyr.

Puntos de datos relevantes:

- Variables clave: edad, sexo, departamento_nom, per_etn_, nom_grupo_, estado, ubicacion, tipo_recuperacion, recuperado, todas las variables de fecha.
- Variables derivadas: dias_diagnostico, dias_recuperacion, dias_muerte.
- Variable de agrupación: sexo(asegúrese de que los valores estén estandarizados, por ejemplo, 'F', 'M').

Resumen: Si sigue los pasos descritos y aborda los posibles desafíos de los datos, podrá realizar un análisis estadístico y visual sólido del conjunto de datos de COVID-19 en R, centrándose en las diferencias demográficas y clínicas por sexo, y visualizar los resultados de manera eficaz en RStudio.

D. Anexo: Continuación del Cuadro 14.– Inicialización automatizada del entorno computacional por el Agente Codificador

```
# Optimized COVID Data Analysis Script for Colombia
# Author: [Your Name]
# Date: [Today's Date]
# Description: Robust, production-ready analysis of COVID data by sex and other variables.
# Feedback-driven improvements for edge cases, missing values, and code maintainability.

# ------------------ Required Libraries ------------------
library(readr)
library(dplyr)
library(tidyr)
library(ggplot2)
library(lubridate)
library(stringr)

# ------------------ Utility Functions -------------------

# Helper to get the actual column name given possible variants
find_colname <- function(df, variants) {
  col <- names(df)[tolower(names(df)) %in% tolower(variants)]
  if (length(col) == 0) return(NA_character_) else return(col[1])
}

# Robust date parser (returns Date or NA)
safe_parse_date <- function(x) {
  # Try common formats; returns NA for invalid
  parse_date_time(x, orders = c("Ymd", "dmY", "mdY", "Y-m-d", "d/m/Y", "m/d/Y"),
                  exact = FALSE, quiet = TRUE)
}

# ------------------ 1. Read & Validate Data ------------------

# Read data
df <- read_csv("covid_colombia.csv", show_col_types = FALSE)
if (nrow(df) == 0) {
  stop("Error: The input file has zero rows.")
}

# Define expected column names (with variants for robustness)
colnames_map <- list(
  sexo = c("sexo", "Sexo", "SEXO"),
  edad = c("edad", "Edad", "EDAD"),
  fecha_inicio_sintomas = c("fecha_inicio_sintomas", "Fecha_inicio_sintomas"),
  fecha_diagnostico = c("fecha_diagnostico", "Fecha_diagnostico"),
  fecha_recuperado = c("fecha_recuperado", "Fecha_recuperado"),
```

```
  fecha_muerte = c("fecha_muerte", "Fecha_muerte"),
  departamento_nom = c("departamento_nom", "Departamento_nom"),
  per_etn_ = c("per_etn_", "Per_etn_"),
  nom_grupo_ = c("nom_grupo_", "Nom_grupo_"),
  estado = c("estado", "Estado"),
  ubicacion = c("ubicacion", "Ubicacion"),
  tipo_recuperacion = c("tipo_recuperacion", "Tipo_recuperacion"),
  recuperado = c("recuperado", "Recuperado")
)

# Get actual column names
actual_cols <- lapply(colnames_map, find_colname, df = df)

# Check for required columns
required_main <- c("sexo", "edad")
missing_main <- required_main[is.na(unlist(actual_cols[required_main]))]
if (length(missing_main) > 0) {
  stop(paste("Error: Required column(s) missing:", paste(missing_main, collapse = ", ")))
}

# ------------------ 2. Data Cleaning & Standardization ------------------

# Rename columns for consistency
for (nm in names(actual_cols)) {
  if (!is.na(actual_cols[[nm]]) && actual_cols[[nm]] != nm) {
    df <- df %>% rename(!!nm := !!sym(actual_cols[[nm]]))
  }
}

# Standardize 'sexo'
df <- df %>%
  mutate(
    sexo = toupper(trimws(as.character(sexo))),
    sexo = ifelse(sexo %in% c("M", "F"), sexo, sexo)
  )
# Warn if unexpected 'sexo' values are present
unexpected_sexo <- setdiff(unique(df$sexo), c("M", "F", NA))
if (length(unexpected_sexo) > 0) {
  warning(sprintf("Warning: Unexpected sexo categories found: %s", paste(unexpected_sexo,
collapse = ", ")))
}

# Convert 'edad' to numeric, warn if non-numeric present
non_numeric_edad <- sum(!is.na(df$edad) & is.na(as.numeric(df$edad)))
if (non_numeric_edad > 0) {
  warning(sprintf("Warning: %d non-numeric 'edad' values set to NA.", non_numeric_edad))
}
df$edad <- as.numeric(df$edad)

# Parse date columns robustly (if present)
for (col in c("fecha_inicio_sintomas", "fecha_diagnostico", "fecha_recuperado",
"fecha_muerte")) {
  if (col %in% names(df)) {
    df[[col]] <- safe_parse_date(df[[col]])
  } else {
    message(sprintf("Column not found: %s", col))
    df[[col]] <- NA
  }
}

# ------------------ 3. Feature Engineering ------------------

df <- df %>%
  mutate(
    dias_diagnostico = as.numeric(fecha_diagnostico - fecha_inicio_sintomas),
    dias_recuperacion = as.numeric(fecha_recuperado - fecha_diagnostico),
    dias_muerte = as.numeric(fecha_muerte - fecha_diagnostico)
  )
```

```
# ------------------ 4. Summary Statistics ------------------

cat("\n--- Edad por Sexo (Resumen estadístico) ---\n")
edad_stats <- df %>%
  group_by(sexo) %>%
  summarise(
    n = sum(!is.na(edad)),
    media = ifelse(n > 0, mean(edad, na.rm = TRUE), NA_real_),
    mediana = ifelse(n > 0, median(edad, na.rm = TRUE), NA_real_),
    min = ifelse(n > 0, min(edad, na.rm = TRUE), NA_real_),
    max = ifelse(n > 0, max(edad, na.rm = TRUE), NA_real_)
  )
print(edad_stats)

# ------------------ 5. Visualizations ------------------

# Plot only if data is present
if (any(!is.na(df$edad)) && length(unique(na.omit(df$sexo))) > 0) {
  ggplot(df, aes(x = edad, fill = sexo)) +
    geom_histogram(position = "identity", alpha = 0.6, bins = 30, na.rm = TRUE) +
    labs(title = "Histograma de Edad por Sexo") +
    theme_minimal() -> p1
  print(p1)

  ggplot(df, aes(x = sexo, y = edad, fill = sexo)) +
    geom_boxplot(na.rm = TRUE) +
    labs(title = "Boxplot de Edad por Sexo") +
    theme_minimal() -> p2
  print(p2)
} else {
  message("Not enough data to plot edad by sexo.")
}

# ------------------ 6. Frequency Tables ------------------

# Helper to print absolute and relative frequency tables
print_freq_tables <- function(varname, by = "sexo") {
  if (varname %in% names(df)) {
    abs_tab <- table(df[[varname]], df[[by]])
    rel_tab <- prop.table(abs_tab, 2)
    cat(sprintf("\n--- Frecuencia absoluta de %s por %s ---\n", varname, by))
    print(abs_tab)
    cat(sprintf("\n--- Frecuencia relativa de %s por %s ---\n", varname, by))
    print(round(rel_tab, 3))
  } else {
    cat(sprintf("Column not found: %s\n", varname))
  }
}

print_freq_tables("departamento_nom")
print_freq_tables("per_etn_")
print_freq_tables("nom_grupo_")

# ------------------ 7. Clinical Time Analysis ------------------

clinical_time_stats <- function(var){
  if (!(var %in% names(df))) {
    cat(sprintf("Variable not found: %s\n", var)); return(invisible(NULL))
  }
  out <- df %>%
    group_by(sexo) %>%
    summarise(
      n = sum(!is.na(.data[[var]])),
      media = ifelse(n > 0, mean(.data[[var]], na.rm = TRUE), NA_real_),
      mediana = ifelse(n > 0, median(.data[[var]], na.rm = TRUE), NA_real_),
      min = ifelse(n > 0, min(.data[[var]], na.rm = TRUE), NA_real_),
      max = ifelse(n > 0, max(.data[[var]], na.rm = TRUE), NA_real_)
```

```
    )
  cat(sprintf("\n--- Estadísticas de %s por sexo ---\n", var))
  print(out)
}
clinical_time_stats("dias_diagnostico")
clinical_time_stats("dias_recuperacion")
clinical_time_stats("dias_muerte")

# Boxplots for clinical times (if data exists)
for (v in c("dias_diagnostico", "dias_recuperacion", "dias_muerte")) {
  if (any(!is.na(df[[v]])) && length(unique(na.omit(df$sexo))) > 0) {
    ggplot(df, aes(x = sexo, y = .data[[v]], fill = sexo)) +
      geom_boxplot(na.rm = TRUE) +
      labs(title = sprintf("Boxplot de %s por Sexo", v)) +
      theme_minimal() -> bp
    print(bp)
  } else {
    cat(sprintf("Not enough data to plot %s by sexo.\n", v))
  }
}

# ------------------ 8. Cross Tables by Sexo ------------------

cross_tab_show <- function(var){
  if (!(var %in% names(df))) {
    cat(sprintf("Column not found: %s\n", var)); return(invisible(NULL))
  }
  abs_tab <- table(df$sexo, df[[var]])
  rel_tab <- prop.table(abs_tab, 1)
  cat(sprintf("\n--- Tabla cruzada absoluta: sexo x %s ---\n", var))
  print(abs_tab)
  cat(sprintf("\n--- Tabla cruzada relativa (por sexo): sexo x %s ---\n", var))
  print(round(rel_tab, 3))
}
cross_tab_show("estado")
cross_tab_show("ubicacion")
cross_tab_show("tipo_recuperacion")
cross_tab_show("recuperado")

# ------------------ 9. Weekly Evolution Plots ------------------

plot_weekly_evolution <- function(date_col, label){
  if (!(date_col %in% names(df))) {
    cat(sprintf("Column not found: %s\n", date_col)); return(invisible(NULL))
  }
  temp <- df %>%
    filter(!is.na(.data[[date_col]]), !is.na(sexo)) %>%
    mutate(semana = floor_date(.data[[date_col]], "week")) %>%
    group_by(semana, sexo) %>%
    summarise(casos = n(), .groups = 'drop')
  if (nrow(temp) == 0) {
    cat(sprintf("No data available for weekly evolution plot: %s\n", label))
    return(invisible(NULL))
  }
  ggplot(temp, aes(x = semana, y = casos, color = sexo)) +
    geom_line() + geom_point() +
    labs(title = paste("Evolución semanal de", label, "por sexo"),
         x = "Semana", y = "Número de casos") +
    theme_minimal() -> p
  print(p)
}
plot_weekly_evolution("fecha_diagnostico", "diagnósticos")
plot_weekly_evolution("fecha_recuperado", "recuperaciones")
plot_weekly_evolution("fecha_muerte", "fallecimientos")
```

E. Anexo: Continuación del Cuadro 16.– Documentación estructural inicial del script: objetivos, funciones y estructura funcional

Documentación
Descripción general
Este script de R realiza un análisis estadístico exhaustivo de los datos de casos de COVID-19 en Colombia, centrándose en las cronologías clínicas y los patrones demográficos por sexo. El flujo de trabajo incluye:

- **Carga y validación de datos robustas** : garantiza que las columnas y los archivos necesarios estén presentes.
- **Transformación de datos** : calcula variables de tiempo clínico clave (por ejemplo, tiempo hasta el diagnóstico, recuperación, muerte).
- **Estadística descriptiva** : Resume y visualiza la edad y los tiempos clínicos por sexo.
- **Análisis categóricos** : genera tablas de frecuencia y tabulaciones cruzadas para variables demográficas y clínicas importantes.
- **Análisis de series de tiempo** : sigue la evolución semanal de casos, recuperaciones y muertes por sexo.
- **Manejo y generación de informes de errores** : utiliza funciones auxiliares para gestionar datos vacíos, columnas faltantes y errores de trazado, lo que garantiza una ejecución fluida incluso con datos incompletos.

El script aprovecha los siguientes paquetes principales: **dplyr** , **ggplot2** , **lubridate** , **readr** , **DT** , **tidyr** , **scales** y **gtsummary** .

Funciones
0. Funciones de utilidad y comprobaciones de paquetes
ensure_packages(pkgs)
Se asegura de que todos los paquetes necesarios estén instalados y cargados. Instala los paquetes faltantes y carga todos los paquetes listados mediante [nombre del archivo] library().

- **Entrada:** Vector de caracteres de nombres de paquetes.
- **Salida:** Ninguna (efectos secundarios: paquetes instalados y cargados).

---

1. Comprobaciones de existencia de archivos y columnas

- **Comprueba la presencia del archivo de datos de entrada** (documentos.csv).
- **Valida que todas las columnas requeridas** estén presentes en el conjunto de datos después de la carga.

---

2. Carga de datos
safely_read_data(file)
Lee de forma segura el conjunto de datos utilizando readr::read_csv()tipos de columnas explícitos y análisis de fechas.

- **Entrada:** Ruta del archivo.

- **Salida:** Marco de datos con columnas y tipos validados.
- **Manejo de errores:** detiene la ejecución con un mensaje claro si faltan el archivo o las columnas, o si falla el análisis.

---

3. Transformación de datos
add_clinical_times(df)
Crea variables de intervalo de tiempo clínico:

- dias_diagnostico:Días desde el inicio de los síntomas hasta el diagnóstico.
- dias_recuperacion:Días desde el diagnóstico hasta la recuperación.
- dias_muerte:Días desde el diagnóstico hasta la muerte.

Cada uno se calcula como la diferencia (en días) entre las columnas de fecha correspondientes:
$dias\_diagnostico_i = fecha\_diagnóstico_i - fecha\_inicio\_sintomas_i$
$dias\_recuperacion_i = fecha\_recuperado_i - fecha\_diagnóstico_i$
$días\_muerte_i = fecha\_muerte_i - fecha\_diagnóstico_i$

- **Entrada:** Marco de datos.
- **Salida:** Marco de datos con nuevas columnas.
- **Controles de calidad:** advierte si alguno de los intervalos es negativo (posibles errores de datos).

---

4. Funciones auxiliares de salida
safe_DT(tbl, caption)
Muestra un marco de datos como una tabla de datos interactiva (a través de DT::datatable) si no está vacío; de lo contrario, imprime un mensaje.
safe_print(x)
Imprime un resumen o una tabla si no está vacío; de lo contrario, imprime un mensaje.
safe_plot(expr)
Evalúa e imprime un gráfico de forma segura, detectando y reportando errores.

---

5. Funciones de análisis modular
5.1.summary_age_by_sex(df)

- Produce una tabla de resumen detallada de las estadísticas de edad (media, DE, mediana, mínimo, máximo, recuento) por sexo utilizando gtsummary::tbl_summaryy dplyr::summarise.
- Muestra los resultados como un resumen formateado y una tabla interactiva.

5.2.plot_age_by_sex(df)

- Dibuja un histograma y un diagrama de caja de distribuciones de edad, facetados y coloreados por sexo.
- Usos ggplot2para la visualización.

5.3.frequency_table_by_sex(df, var, caption)

- Genera una tabla de frecuencias (recuentos) para una variable categórica dada, tabulada de forma cruzada por sexo.
- Elimina los valores faltantes.
- Muestra la tabla de forma interactiva.

5.4.clinical_time_analysis_by_sex(df)

- Resume y visualiza los intervalos de tiempo clínicos (dias_diagnostico, dias_recuperacion, dias_muerte) por sexo.
- Produce resúmenes estadísticos y diagramas de caja para cada intervalo.
- Maneja los datos faltantes de forma adecuada (por ejemplo, solo los que no faltan para cada intervalo).

5.5.cross_tab_by_sex(df, var, caption)

- Un contenedor para frequency_table_by_sex, específicamente para variables categóricas clínicas.

5.6.weekly_evolution_plot(df, date_col, ylab, title)

- Agrega eventos por semana y sexo para una columna de fecha determinada (por ejemplo, diagnóstico, recuperación, muerte).
- Traza los recuentos semanales como un gráfico lineal, coloreado por sexo, con ejes de fecha formateados.

6. Ejecución del análisis

- **Secuencia de análisis:** si hay datos presentes, ejecuta todos los análisis de resumen, visualización, tabulación y series de tiempo en orden.
- **Cada paso es resistente a datos faltantes o vacíos** y los resultados están claramente etiquetados.

7. Notas finales

- Imprime un mensaje de finalización, recordando a los usuarios que deben verificar si hay advertencias o errores.

Transferencias de agentes (pseudocódigo)

Este script es autónomo y no requiere transferencias de agentes. Sin embargo, su estructura modular facilita su integración en un flujo de trabajo más amplio. Un flujo de trabajo típico (pseudocódigo) es:

```
# Pseudocode for agent interactions

Ensure all required R packages are installed and loaded.
If input file is missing, stop with error.
Load and validate data, ensuring required columns.
Compute clinical time intervals (dias_diagnostico, dias_recuperacion, dias_muerte).
For each analysis step:
    - If data is available for the analysis:
        - Run the summary or visualization function.
        - Display output (table or plot).
At end, print completion message.
```

Ecuaciones

Intervalos de tiempo clínicos

Los intervalos de tiempo clínicos se calculan como diferencias (en días) entre fechas relevantes:

Días hasta el diagnóstico

dias_diagnosticoi=fecha_diagnósticoi−fecha_inicio_sintomasidias_diagnostico*i* =fecha_diagnóstico*i*−fecha_inicio_sintomas*i*

Días para la recuperación

dias_recuperacioni=fecha_recuperadoi−fecha_diagnósticoidias_recuperacion*i*=fecha_recuperado*i* −fecha_diagnóstico*i*

Días hasta la muerte

días_muertei=fecha_muertei−fecha_diagnósticoidías_muerte*i*=fecha_muerte*i*−fecha_diagnóstico*i*

Donde todas las variables de fecha están en Dateformato R, por lo que la resta da como resultado el intervalo en días.

Una ecuación en línea podría verse así:incógnita2+y2=z2*incógnita*2+*y*2=*z*2.

Notas

- **Manejo de errores:** el script está diseñado para manejar datos faltantes, vacíos o inconsistentes en cada paso, lo que garantiza una salida sólida y comentarios del usuario.
- **Extensibilidad:** Las funciones modulares permiten una fácil extensión a nuevas variables o factores de agrupación.
- **Visualización:** Todas las tablas y gráficos están etiquetados y son interactivos siempre que sea posible, lo que facilita la exploración y la presentación.
- **Reproducibilidad:** la gestión explícita de paquetes y las comprobaciones de columnas garantizan que los análisis sean reproducibles y portátiles.

Esta documentación debería proporcionar una guía completa para los usuarios y desarrolladores que trabajan con el código R proporcionado para el análisis de datos epidemiológicos.

Su pregunta sobre la documentación o el código:

F. Anexo: Resultados gráficos y tabulares generados por el Asistente Multiagente IA en R

```
> print(edad_stats)
# A tibble: 2 × 6
  sexo      n media mediana   min   max
  <chr> <int> <dbl>   <dbl> <dbl> <dbl>
1 F       440  43.9      40     2   100
2 M       560  42.1      38     1    95

> print_freq_tables("departamento_nom")

--- Frecuencia absoluta de departamento_nom por sexo ---

                   F   M
  ANTIOQUIA       94 132
  ARAUCA           3   0
  ATLANTICO        6   4
  BARRANQUILLA    25  30
  BOGOTA          52 124
  BOLIVAR          2   6
  BOYACA           1   0
  CALDAS           6   6
  CAQUETA          1   1
  CARTAGENA       14  38
  CASANARE         2   2
  CAUCA            2   8
  CESAR            0   2
  CORDOBA          6   5
  CUNDINAMARCA    10   7
  GUAJIRA          5   7
  HUILA            5   1
  MAGDALENA        1   0
  META             9   8
  NARIÑO           1   3
  NORTE SANTANDER  3   4
  QUINDIO          2   2
  RISARALDA        4   8
  SANTANDER       36  75
  STA MARTA D.E.   4   9
  SUCRE           11  14
  TOLIMA          20   6
  VALLE          115  58

--- Frecuencia relativa de departamento_nom por sexo ---

                     F     M
```

```
  ANTIOQUIA       0.214 0.236
  ARAUCA          0.007 0.000
  ATLANTICO       0.014 0.007
  BARRANQUILLA    0.057 0.054
  BOGOTA          0.118 0.221
  BOLIVAR         0.005 0.011
  BOYACA          0.002 0.000
  CALDAS          0.014 0.011
  CAQUETA         0.002 0.002
  CARTAGENA       0.032 0.068
  CASANARE        0.005 0.004
  CAUCA           0.005 0.014
  CESAR           0.000 0.004
  CORDOBA         0.014 0.009
  CUNDINAMARCA    0.023 0.013
  GUAJIRA         0.011 0.013
  HUILA           0.011 0.002
  MAGDALENA       0.002 0.000
  META            0.020 0.014
  NARIÑO          0.002 0.005
  NORTE SANTANDER 0.007 0.007
  QUINDIO         0.005 0.004
  RISARALDA       0.009 0.014
  SANTANDER       0.082 0.134
  STA MARTA D.E.  0.009 0.016
  SUCRE           0.025 0.025
  TOLIMA          0.045 0.011
  VALLE           0.261 0.104

> print_freq_tables("per_etn_")

--- Frecuencia absoluta de per_etn_ por sexo ---

     F   M
  1   7   6
  5  19  25
  6 414 529

--- Frecuencia relativa de per_etn_ por sexo ---

        F     M
  1 0.016 0.011
  5 0.043 0.045
  6 0.941 0.945
> print_freq_tables("nom_grupo_")

--- Frecuencia absoluta de nom_grupo_ por sexo ---

             F M
  EMBERA     1 1
  PIJAO      1 0
  Por definir 3 5
  WAYUU      1 0
  ZENU       1 0

--- Frecuencia relativa de nom_grupo_ por sexo ---

                 F     M
  EMBERA      0.143 0.167
  PIJAO       0.143 0.000
  Por definir 0.429 0.833
  WAYUU       0.143 0.000
  ZENU        0.143 0.000

> clinical_time_stats("dias_diagnostico")

--- Estadísticas de dias_diagnostico por sexo ---
# A tibble: 2 × 6
```

```
  sexo      n   media mediana   min     max
  <chr> <int>   <dbl>   <dbl> <dbl>   <dbl>
1 F       439 963193. 1209600     0 2678400
2 M       549 986439. 1209600     0 2246400

> clinical_time_stats("dias_recuperacion")

--- Estadísticas de dias_recuperacion por sexo ---
# A tibble: 2 × 6
  sexo      n media mediana   min   max
  <chr> <int> <dbl>   <dbl> <dbl> <dbl>
1 F       422  20.8      11     2   291
2 M       538  24.4      12     2   337
> clinical_time_stats("dias_muerte")

--- Estadísticas de dias_muerte por sexo ---
# A tibble: 2 × 6
  sexo      n    media mediana     min      max
  <chr> <int>    <dbl>   <dbl>   <dbl>    <dbl>
1 F        19 1568842.  604800       0  7344000
2 M        23 2024765.  691200 -518400 26784000
> cross_tab_show("estado")

--- Tabla cruzada absoluta: sexo x estado ---

    Fallecido Leve N/A
  F        16  421   3
  M        19  537   4

--- Tabla cruzada relativa (por sexo): sexo x estado ---

    Fallecido  Leve   N/A
  F     0.036 0.957 0.007
  M     0.034 0.959 0.007
> cross_tab_show("ubicacion")

--- Tabla cruzada absoluta: sexo x ubicacion ---

    Casa Fallecido N/A
  F  421        16   3
  M  537        19   4

--- Tabla cruzada relativa (por sexo): sexo x ubicacion ---

     Casa Fallecido   N/A
  F 0.957     0.036 0.007
  M 0.959     0.034 0.007

> cross_tab_show("tipo_recuperacion")

--- Tabla cruzada absoluta: sexo x tipo_recuperacion ---

    PCR Tiempo
  F 104    318
  M 134    404

--- Tabla cruzada relativa (por sexo): sexo x tipo_recuperacion ---

      PCR Tiempo
  F 0.246  0.754
  M 0.249  0.751

> cross_tab_show("recuperado")

--- Tabla cruzada absoluta: sexo x recuperado ---
```

```
    Fallecido N/A Recuperado
  F        16   2        422
  M        19   3        538

--- Tabla cruzada relativa (por sexo): sexo x recuperado ---

    Fallecido   N/A Recuperado
  F     0.036 0.005      0.959
  M     0.034 0.005      0.961
> plot_weekly_evolution <- function(date_col, label){
> plot_weekly_evolution("fecha_diagnostico", "diagnósticos")
> plot_weekly_evolution("fecha_recuperado", "recuperaciones")
> plot_weekly_evolution("fecha_muerte", "fallecimientos")
```